FASTER3D: EXPLORING THE ROLES OF LATENT TOKENS FOR FASTER IMAGE-TO-3D GENERATION

Anonymous authors
Paper under double-blind review



Figure 1: **Faster3D** accelerates image-to-3D generation across five backbones while largely preserving geometry and visual details. Red labels indicate speedups in shape generation.

ABSTRACT

Flow-based image-to-3D models repeatedly predict velocities for all latent tokens during sampling, although the need for recomputation can vary across tokens. Image-to-3D generation is primarily concerned with the sparsity problem that the surrounding space of the formed object always remains empty. Our analysis of TRELLIS sparse-structure generation shows that *non-surface* tokens associated with empty regions generally have smaller velocity magnitudes and directional changes than *surface* tokens associated with occupied regions. These observations motivate selective velocity reutilization. However, omitting selected tokens from self-attention can degrade geometry because their representations still provide context for other tokens. In this paper, we propose **Faster3D**, a *training-free* acceleration framework with two components. The *Temporal Token Decoupling* which instructs the reutilization of velocity predictions, and the *Cached Context Restoration* retains cached keys/values at their original positions for sampling. These two components reduce redundant velocity computation while preserving the attention context provided by selected tokens. Experiments on the Toys4K dataset show that **Faster3D** consistently accelerates diverse image-to-3D backbones by $1.30\times$–$1.95\times$ speedups in the targeted structure or geometry sampling stage, with geometry and appearance metrics remaining close to the full-compute baselines. These results highlight context-preserving token-wise velocity reutilization as an effective technique for image-to-3D accelerations.

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.

1 INTRODUCTION

Flow-based generative models have substantially advanced image-conditioned 3D generation. Representative image-to-3D models, including TRELLIS (Xiang et al., 2025), TRELLIS.2 (Xiang et al., 2026), Hunyuan3D-2.1 (Hunyuan3D et al., 2025), and Step1X-3D (Li et al., 2025b), generate high-quality 3D assets from a input image. Their geometry generators are commonly built on Diffusion Transformer (DiT) (Peebles & Xie, 2023) architectures and often follow a coarse-to-fine pipeline. They commonly first establish the underlying 3D structure and then synthesize the fine-grained geometry and appearance. Nevertheless, this generation process involves time-consuming iterative inferences. Specifically, at each sampling step, the DiT predicts velocities per latent token, even while the majority of the emerging 3D structure become gradually stable.

To accelerating 3D generation, existing approaches adopt training-based distillation (Lee et al., 2026; Liu et al., 2025c; Yang et al., 2025) or training-free inference-time optimization (Feng et al., 2026; Yang et al., 2026). In training-based regime, MDT-dist (Zhou et al., 2026) and FlashVDM (Lai et al., 2025) reduce sampling steps through few-step flow distillation and progressive flow distillation. However, these training-based methods generally requires large-scale training computations. In contrast, *training-free* methods exploit computational redundancy in pretrained generators to accelerate inference while avoiding additional training overhead. For example, Fast3Dcache (Yang et al., 2026) adapts caching budgets to geometric stabilization and selects tokens for reuse based on velocity and acceleration. Fast-SAM3D (Feng et al., 2026) combines modality-aware caching for shape and layout, token reduction for texture refinement, and adaptive aggregation for mesh decoding. Nevertheless,these methods exploit geometric or temporal stability to reduce computation. We investigate a complementary question: *when a token's velocity can be reused, should its representation also be removed from the self-attention context available to other tokens?*

We begin with the sparse-structure generation stage of TRELLIS, whose latent tokens lie on a regular 3D grid. For analysis, we label a token as surface-associated if its corresponding block in the final decoded occupancy grid contains at least one occupied voxel, and as non-surface-associated otherwise. We refer to them as *surface* tokens and *non-surface* tokens, respectively. These retrospective labels describe coarse spatial correspondence and are unavailable during inference. Current standard sampling approaches often treat all tokens the same at every step for computing. This motivates a critical thinking about *whether the non-surface tokens contribute the same as those surface tokens during sampling.* Figure 2 provides a clue: At the first sampling step, blending the current latent and the velocity prediction can already derive an initial estimation for the latent representation of the final 3D structure. We feed the prediction directly into the sparse-structure decoder and the texture stage, and obtain similar final outputs as the full-computation backbones. Comparing the result with the full backbone output leads to the following observation: Predictions from early sampling stages provide provisional indications of the emerging geometry and offers an initial indication for the spatial roles of latent tokens.



Figure 2: 3D assets generated by using the geometry predicted from the first step in shape stage, as shown in the middle, already obtain similar final output as the fully computed baseline (right).

Inspired by training-free diffusion acceleration methods (Li et al., 2024; Ma et al., 2024), we first examine how the predicted velocities of *surface* and *non-surface* tokens evolve during sampling. From our analysis, we find that surface tokens generally remain more active. Meanwhile, many non-surface tokens exhibit stable velocities and are candidates for velocity reutilization. However, directly excluding those non-surface tokens by reusing velocity predictions is prone to generate incomplete geometries. Yet, these non-surface tokens can provide meaningful self-attention context for the surface tokens.

Motivated by the above findings, we propose **Faster3D**, a training-free approach for image-to-3D acceleration. **Faster3D** is composed of two core novel techniques, the first technique named *Temporal Token Decoupling (TTD). TTD* computes a token activity score from the latent prediction and uses a time-dependent threshold to select tokens for velocity reutilization. Selected tokens reuse their previous velocity predictions, while the remaining tokens receive new predictions from the

DiT. The second technique named *Cached Context Restoration (CCR)*. *CCR* restores the cached keys and values for such selected tokens at their original positions. These cached states complement the fresh computed keys and values of the remaining tokens to preserve the full self-attention context. Periodic full-token computations refresh both the cached velocities and the keys and values. Together, *TTD* and *CCR* reduce redundant computation while keeping geometry and texture quality close to the backbone outputs, without additional training.

Experiments on diverse image-to-3D base models demonstrate faster structure generation while largely preserving full-compute quality. On diverse typical image-to-3D backbones, **Faster3D** outperforms existing 3D acceleration approach (Yang et al., 2026) by demonstrating more detail-enhanced generations, more intact geometries, and higher acceleration ratios. Our main contributions are summarized as follows:

- We identify and analyze the roles of latent tokens in 3D geometry generation. Our analysis shows that *non-surface* tokens mostly exhibit stable velocities than *surface* tokens during flow sampling. Their velocity predictions can be reutilized, while they still provide important self-attention context for the remaining active tokens.
- We introduce **Faster3D**, a training-free framework with two components: *Temporal Token Decoupling(TTD)* and *Cached Context Restoration(CCR)* . *TTD* uses activity from the latent prediction to select tokens for velocity reutilization. *CCR* restores their cached keys and values at their original token positions to preserve full self-attention context.
- Extensive experiments demonstrate that **Faster3D** accelerates diverse image-to-3D backbones while largely preserving the geometry and texture quality. Our method **Faster3D** achieves better generation quality than existing methods at comparable inference speed, demonstrating the benefits of velocity reutilization with preserved self-attention context.

2 Related Work

2.1 Diffusion and Flow-Based 3D Generative Models

Diffusion models (Ho et al., 2020) generate samples through iterative denoising processes, while flow matching (Lipman et al., 2023) and rectified-flow models (Liu et al., 2023) learn vector fields that transport noise toward the data distribution. Building on these advances, native 3D generative models learn compact 3D latent representations and perform generation within this latent space. Recent image-to-3D systems, including TRELLIS (Xiang et al., 2025), TRELLIS.2 (Xiang et al., 2026), and Hunyuan3D-2.1 (Hunyuan3D et al., 2025), adopt multi-layer Transformer backbones to generate high-quality geometry and appearance. Notably, TRELLIS introduces a Structured Latent Representation (SLAT) that supports decoding into meshes, radiance fields, and 3D Gaussians. Despite their strong generation capabilities, these models require repeated computations during inference time. In this paper, we build **Faster3D** on these pretrained 3D generators and reduces redundant during geometry generation, without changing their pretrained weights.

2.2 Training-Free Acceleration for Visual Generation

Training-free acceleration reduces inference computation without updating pretrained model parameters. In image and video generation, existing methods reuse intermediate features across sampling steps (Ma et al., 2024; Selvaraju et al., 2024), adapt caching schedules (Liu et al., 2025a; Kahatapitiya et al., 2025), or use feature extrapolation and token-wise computation allocation (Liu et al., 2025b; Zou et al., 2025). Beyond diffusion and flow models, FasterVAR (Li et al., 2026) exploits stage-dependent redundancy in visual autoregressive generation, preserving early semantic and structural stages while approximating later fidelity refinement. In 3D generation, Fast3Dcache (Yang et al., 2026) adjusts caching budgets according to geometric stabilization and selects tokens for reuse based on velocity and acceleration. However, our experiments show that some assets generated with Fast3Dcache exhibit incomplete geometry. Fast-SAM3D (Feng et al., 2026) combines modality-aware caching, token reduction for texture refinement, and adaptive token aggregation for mesh decoding. Our work focuses on the roles of latent tokens in 3D geometry generation. Specifically, **Faster3D** uses token activity to guide velocity reutilization and restores cached keys and values at their original token positions to preserve full self-attention context.

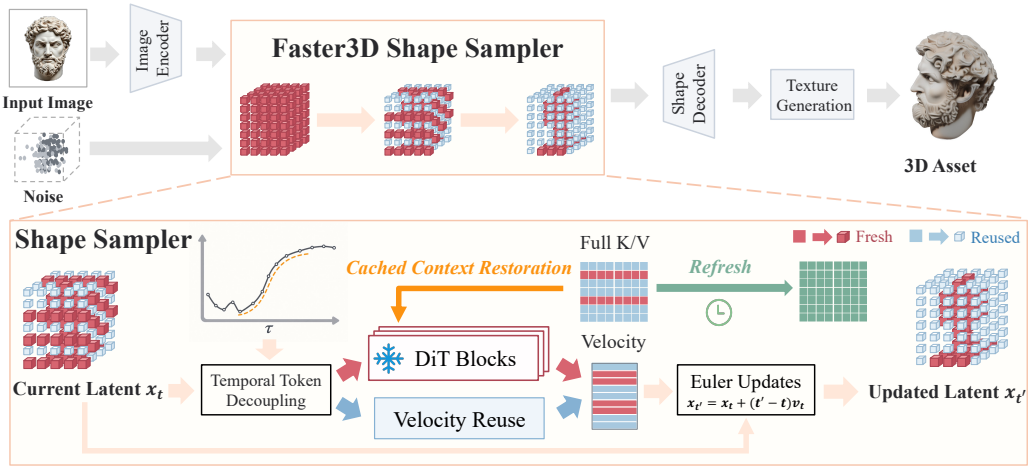


Figure 3: Overview of **Faster3D**. **Top:** The complete image-to-3D pipeline with **Faster3D** sampling and unchanged texture stages. **Bottom:** Details of the **Faster3D** shape sampler. *Temporal Token Decoupling (TTD)* uses token activity and a fitted threshold to select tokens for *velocity reutilization*. Within the DiT, *Cached Context Restoration (CCR)* restores cached keys and values at their original token positions to preserve full self-attention context. Euler updates use fresh and reused velocities to update all latent tokens. Caches are periodically refreshed during sampling.

3 METHODOLOGY

In this section, we first establish the preliminaries related to flow models and image-to-3D models in Sec 3.1. Following, we detail our proposed **Faster3D** in Sec 3.2 and Sec 3.3, corresponding to our *Temporal Token Decoupling* and *Cached Context Restoration* technique respectively.

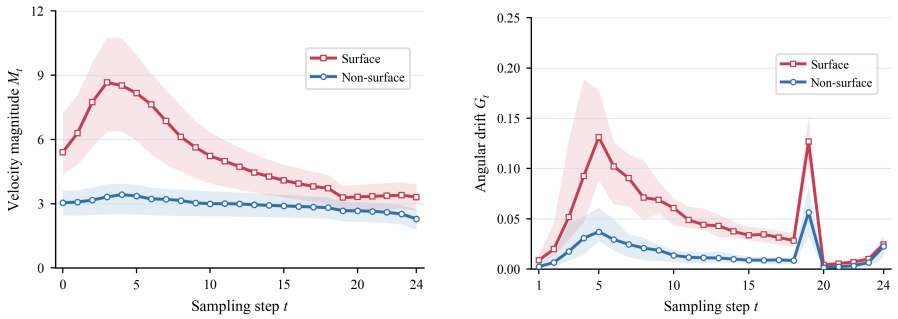
3.1 PRELIMINARIES

Flow-based 3D generation. We use TRELLIS (Xiang et al., 2025) as the primary backbone to introduce **Faster3D** in this section. Its pipeline comprises sparse-structure generation (Stage I) and structured latent (SLAT) generation (Stage II). In Stage I, a 3D-VAE represents a 64^3 binary occupancy grid as a dense latent tensor $\mathbf{x} \in \mathbb{R}^{8\times16\times16\times16}$. The DiT processes its 4,096 spatial tokens and predicts a velocity tensor of the same shape, conditioned on the input image. The structure decoder maps the sampled latent back to occupancy logits, whose positive locations define the support for Stage II. Faster3D modifies Stage-I sampling while leaving Stage II and asset decoding unchanged.

Following flow matching (Lipman et al., 2023), It uses the path $\mathbf{x}_t = (1-t)\mathbf{x}_0 + \sigma(t)\boldsymbol{\epsilon}$, where $\mathbf{x}_0$ is the clean latent, $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$, and $\sigma(t) = \sigma_{\min} + (1-\sigma_{\min})t$ with minimum noise scale $\sigma_{\min}$. The DiT predicts token-wise velocities from the current latent, timestep, and image features. The sampler uses $\mathbf{v}_t = (1+s)\mathbf{v}_t^{\text{cond}} - s\mathbf{v}_t^{\text{uncond}}$, where the superscripts denote conditional and unconditional predictions, and s is the CFG guidance strength. Sampling proceeds from $t = 1$ to $t = 0$ through Euler updates $\mathbf{x}_{t'} = \mathbf{x}_t + (t'-t)\mathbf{v}_t$, where $t' < t$. The current state and velocity yield a final latent estimate $\widehat{\mathbf{x}}_{0,t} = (1-\sigma_{\min})\mathbf{x}_t - \sigma(t)\mathbf{v}_t$, which we term as the *latent prediction*.

Sparse-structure generation. Stage I samples a dense latent grid, which is decoded into occupancy logits. Thresholding these logits at zero produces a binary occupancy grid whose occupied locations are used for Stage II. For the analysis below, surface and non-surface token labels are derived retrospectively from the final decoded structure and are not available during sampling.

K/V cache. Self-attention aggregates values according to query–key similarities. The K/V caching technique (Tian et al., 2024; Zhang et al., 2023; Xiao et al., 2024) stores previously computed keys and values at each attention layer so that subsequent queries can attend to them without recomputation. These cached states provide attention context but may become outdated as the underlying latent tokens evolve.



(a) Velocity magnitude M_t. (b) Directional drift G_t.

Figure 4: **Velocity dynamics.** *Non-surface* tokens exhibit lower velocity magnitudes and less directional changes than *surface* tokens.

3.2 Temporal Token Decoupling

In this section, we detail our proposed *Temporal Token Decoupling (TTD)*, *TTD* can gradually decouple tokens in latent prediction during sampling.

Analysis of token dynamics. Reusing previously computed results is a common strategy for training-free acceleration (Lyu et al., 2025; Zhao et al., 2025; Cao et al., 2026). Iterative sampling involves repeated network evaluations, but some intermediate features change marginally across adjacent steps and do not require recomputation at every sampling step. Previous methods, including DeepCache (Ma et al., 2024) and Faster Diffusion (Li et al., 2024), empirically observe this temporal stability and reuse intermediate features to reduce redundant computation.

Inspired by above, we experimentally examine the temporal dynamics of predicted velocities for *surface* and *non-surface* tokens during image-to-3D sampling. We define the velocity magnitude as: $M_t(i) = \|\mathbf{v}_t(i)\|_2$ and the adjacent-step directional variation as:

$$G_t(i) = \begin{cases} 1 - \dfrac{\langle \mathbf{v}_t(i), \mathbf{v}_{t-1}(i) \rangle}{M_t(i) M_{t-1}(i)}, & \min(M_t(i), M_{t-1}(i)) > \eta, \\ 0, & \max(M_t(i), M_{t-1}(i)) \le \eta, \\ 1, & \text{otherwise}. \end{cases} \tag{1}$$

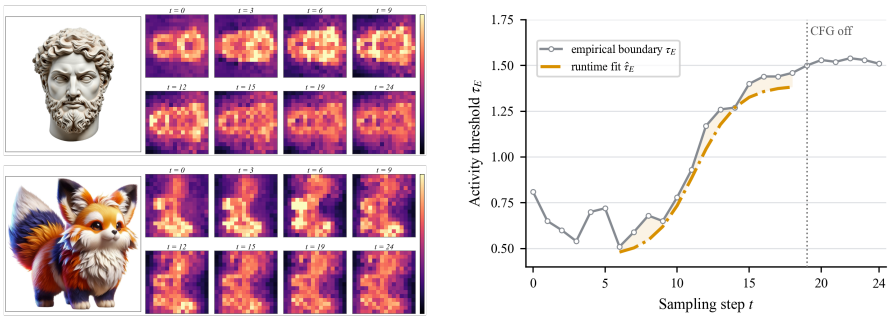
Here, t indexes sampling steps, i indexes latent tokens, $\mathbf{v}_t(i)$ denotes the predicted velocity of token i at step t, and $\eta = 10^{-8}$ is the numerical cutoff. We assign zero directional variation when both velocities are below this cutoff and use one as an explicit convention when only one is below it. Lower $G_t(i)$ indicates less directional change between adjacent steps.

We compute $M_t(i)$ and $G_t(i)$ during full-compute TRELLIS sparse-structure sampling. Token labels follow the definition above, using the final decoded structure and a $4 \times 4 \times 4$ voxel block per token. These labels are used only for offline analysis and calibration. Figure 4 shows that *non-surface* tokens generally exhibit lower velocity magnitudes and smaller directional changes than *surface* tokens. These observations suggest that many *non-surface* tokens may be suitable for velocity reutilization, reducing repeated velocity computation.

Token activity score. Because final occupancy labels are unavailable during sampling, we use the standardized magnitude of the clean-latent prediction as a lightweight selection proxy. We refer to this quantity as the *token activity score*. This score measures deviation from the sample-wise channel means; it is not a direct measure of temporal change. Its relevance to velocity reuse is assessed empirically through the trajectory analysis. The token activity score from the latent prediction is:

$$A_t(i) = \left[\sum_{c=1}^{C} \left(\frac{\widehat{\mathbf{x}}_{0,t}(c,i) - \mu_{t,c}}{s_{t,c} + \varepsilon} \right)^2 \right]^{1/2}, \tag{2}$$

where C is the number of latent channels, $\mu_{t,c}$ and $s_{t,c}$ are the mean and population standard deviation of channel c across all tokens within each sample, and $\varepsilon = 10^{-6}$. Channel-wise standardization



(a) Spatial evolution of token activity. (b) Fitted activity boundary.

Figure 5: **Activity proxy and temporal calibration.** (a) Spatial slices show A_t concentrating around the emerging geometry. (b) A conservative logistic curve fits the step-wise activity boundaries obtained on the calibration set. Final geometry is used only for offline calibration; inference uses A_t and the fixed fitted curve without runtime decoding.

reduces sensitivity to channel offsets and scales, supporting a shared threshold across assets. During selective-computation steps, $\mu_{t,c}$ and $s_{t,c}$ are computed from the current latent prediction. Figure 5-(a) shows that high activity concentrates around the emerging geometry, motivating low-activity tokens as candidates for velocity reutilization.

Threshold calibration. To determine the shared threshold across general image-to-3D assets, we analyze conservative step-wise activity using retrospective surface labels from full-compute outputs of a calibration set. These thresholds limit surface-token admission and vary across sampling steps. We fit their temporal trend with a logistic curve $\widehat{\tau}_E(t)$ and adjust it downward to remain below the calibrated thresholds, as shown in Figure 5-(b). The curve is fitted offline and fixed during sampling, so token selection requires neither online fitting nor intermediate decoding.

To balance quality and efficiency, we expand this conservative boundary by a factor α. A freshly evaluated token satisfying $A_t(i) \leq \alpha\widehat{\tau}_E(t)$ becomes eligible for velocity reutilization in subsequent steps. Eligibility is determined only after a fresh DiT prediction and takes effect at the next sampling step. Reused velocities do not count as new evidence of stability. Global refreshes recompute all tokens and reassess eligibility. Larger α broadens the candidate set and may include low-activity surface tokens. Offline analysis suggests that these additional surface tokens have relatively stable velocities and small velocity reutilization errors. Selection follows token activity, while all latent tokens continue to undergo Euler updates. Further details are provided in Appendix C.1.

3.3 Cached Context Restoration

Although *TTD* can select tokens for velocity reutilization to accelerate the image-to-3D generation, directly excluding these tokens from DiT computation also removes their keys/values from self-attention and thus leads to incomplete geometries. Similar manipulations and problems are also observed in existing image-to-3D acceleration practice Fast3Dcache (Yang et al., 2026). To address this issue, we introduce *Cached Context Restoration* as below to preserve this context without recomputing the selected tokens at *velocity reutilization* steps.

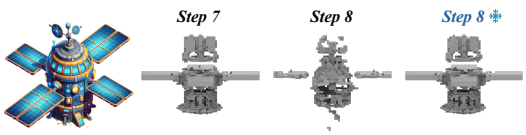


Figure 6: **The effect of cached K/V restoration.** From left to right: input image, step-7 predicted structure, step-8 predictions without and with K/V restoration.

As prior works (Tian et al., 2024; Zhang et al., 2023; Xiao et al., 2024) suggest, KV caching may help preserve the attention context of selected tokens during velocity reutilization. Beyond diffusion models, visual autoregressive models (Tian et al., 2024) use KV caching to avoid recomputing earlier-scale representations while retaining their attention context for subsequent generation (Han et al., 2025; Qin et al., 2026; Li et al., 2025a).

To verify the effectiveness of K/V caching in image-to-3D generation, we compare next-step predictions with and without cached K/V restoration, as shown in Figure 6. We keep the current latent

state, the set of tokens selected for velocity reutilization, and the reused velocities fixed for step 8 in this 25-step generation pipeline. We find that restoration of cached K/V can alleviate distortion in the decoded structure and reduce velocity prediction errors on active tokens. More proof experiments are shown in Appendix C.3. All these results support preserving attention context during velocity reutilization, and we intuitively propose the *Cached Context Restoration (CCR)* as below.

Original-position K/V restoration. *CCR* maintains a token-wise K/V cache at the self-attention layer of each DiT block. At sampling step t, let $\mathcal{R}_t$ denote the indices of tokens selected for velocity reutilization and $\mathcal{F}_t$ the remaining token indices. For DiT block ℓ, we combine newly computed and cached K/V states at their original token positions:

$$\left(\overline{\mathbf{K}}_{t,i}^{(\ell)}, \overline{\mathbf{V}}_{t,i}^{(\ell)}\right) = \begin{cases} \left(\mathbf{K}_{t,i}^{(\ell)}, \mathbf{V}_{t,i}^{(\ell)}\right), & i \in \mathcal{F}_t, \\ \left(\mathbf{K}_{\rho_t(i),i}^{(\ell)}, \mathbf{V}_{\rho_t(i),i}^{(\ell)}\right), & i \in \mathcal{R}_t, \end{cases} \tag{3}$$

where i indexes latent tokens and $\rho_t(i)$ denotes the most recent step at which token i received a new DiT prediction. Queries computed for $\mathcal{F}_t$ attend to this complete K/V context. Thus, selected tokens bypass DiT computation while their cached keys and values continue to provide attention context for the remaining tokens. Newly computed K/V states overwrite the corresponding cache entries, while the other entries are retained.

Cache refresh scheduling. All latent tokens evolve through Euler updates, so cached K/V states and velocity predictions can become outdated. Our offline analysis suggests that elapsed steps since the latest global refresh are informative about velocity prediction errors from context reuse on active tokens. We therefore use elapsed steps as a low-cost proxy for cache lifespan and compute $H_t^{\text{prop}} = H + \min((t-r)/K, 1)$, where t indexes sampling steps, r is the latest global refresh step, H is the accumulated score, and K is the maximum refresh interval. A refresh is triggered when $t - r \geq K$, or when $t - r \geq K_{\min}$ and $H_t^{\text{prop}} > \delta$.

Each refresh recomputes all tokens, updates both the K/V and velocity caches, reevaluates token activity, and resets $H = 0$ and $r = t$. Otherwise, we set $H \leftarrow H_t^{\text{prop}}$ and continue selective computation with cached K/V restoration. Cache initialization and the CFG transition also require full-token computation. More details about this mechanism are provided in Appendix C.2.

Overall, our acceleration framework **Faster3D**, composing of *Temporal Token Decoupling (TTD)* for velocity reutilization and *Cached Context Restoration (CCR)* for cached K/V restoration, reduce redundant computation while generate complete geometry.

4 Experiments

We describe the experimental setup and evaluation protocol in Sec 4.1. We then evaluate quality and efficiency across five 3D backbones in Sec 4.2 and analyze *Temporal Token Decoupling (TTD)* and *Cached Context Restoration (CCR)* through ablations in Sec 4.3.

4.1 Experimental Setup

Evaluation Benchmark. We evaluate our method Faster3D on Toys4K (Stojanov et al., 2021) to compare with comparison methods Fast3Dcache Yang et al. (2026). The comparison is conducted on 210 objects, composing of two objects per category across 105 categories. For each rendered input view per object, we have the corresponding ground-truth assets for geometry and appearance evaluation. Additional occupancy comparisons are provided in Appendix C.

Evaluation Metrics. We report vIoU (Choy et al., 2016), Chamfer Distance (CD) (Barrow et al., 1977; Pumarola et al., 2020), and F-score (Tatarchenko et al., 2019) for geometry; PSNR and LPIPS (Zhang et al., 2018) for rendered appearance; and ULIP-2 (Xue et al., 2024) and Uni3D (Zhou et al., 2024) for image–3D semantic consistency. All runs use a single NVIDIA RTX PRO 6000 Blackwell GPU, with efficiency measured by stage-wise wall-clock speedup and model-core FLOPs (Hernandez & Brown, 2020; Yang et al., 2026) in TFLOPs. Implementation and metric details are provided in Appendix B.

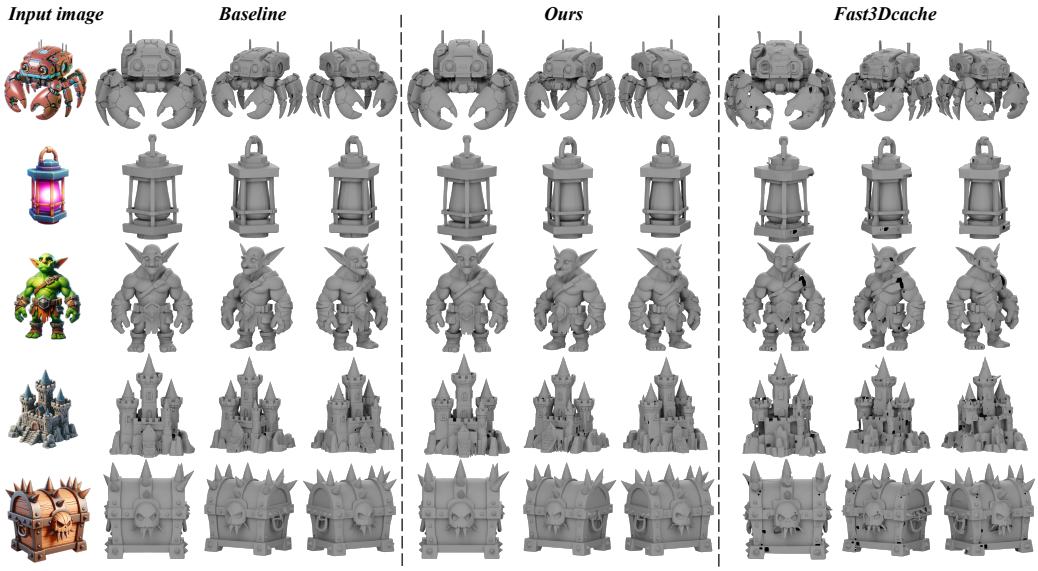


Figure 7: **Qualitative geometric comparison on TRELLIS.** Each row shows an input image followed by full-compute TRELLIS, **Faster3D**, and Fast3Dcache, each rendered from three identical viewpoints using surface normals. Transfer results on other backbones are provided in Appendix D..

Table 1: Quantitative comparison and training-free transfer on Toys4K-210. Speedups and FLOPs refer to the targeted sampling stage. † marks our Fast3DCache adaptations. All values are consistently rounded from the underlying high-precision results. Their gray semantic scores are excluded from ranking and heatmap normalization. **Bold**/underline indicate the best/second-best raw-value ranks within each backbone, including unhighlighted baselines. Colors indicate relative performance among comparable entries, from blue (worse) to pink (better).

Method	Efficiency		Appearance		Geometry			Semantic Alignment	
	Speedup↑ (×)	FLOPs↓ (T)	PSNR↑ (dB)	LPIPS↓	vIoU↑ (%)	CD↓	F-score↑ (%)	ULIP-2↑	Uni3D↑
TRELLIS									
Baseline	1.00	244.24	20.66	0.096	61.03	0.0278	95.93	0.4607	0.4068
w/ Fast3DCache	1.338	**132.43**	20.27	0.101	57.38	0.0284	95.93	0.4185	0.3726
w/ **Faster3D**	**1.342**	138.03	**20.64**	**0.097**	**60.65**	**0.0278**	**95.99**	**0.4613**	**0.4091**
TRELLIS.2									
Baseline	1.00	303.25	20.76	0.096	68.62	0.0224	97.37	0.4420	0.3931
w/ Fast3DCache†	1.24	223.38	20.81	0.096	67.24	0.0228	97.28	0.4377	0.3881
w/ **Faster3D**	**1.34**	**215.15**	**20.89**	**0.095**	**68.59**	**0.0224**	**97.31**	**0.4395**	**0.3912**
Hunyuan3D-2.1									
Baseline	1.00	1824.16	20.67	0.094	66.82	0.0199	98.23	0.4520	0.4046
w/ Fast3DCache†	1.76	1009.96	20.67	0.095	65.95	0.0203	98.19	0.4498	0.4017
w/ **Faster3D**	**1.95**	**861.60**	**20.68**	**0.094**	**67.34**	**0.0200**	**98.21**	**0.4514**	**0.4021**
TripoSG									
Baseline	1.00	627.02	–	–	58.31	0.0314	94.37	0.4315	0.3806
w/ **Faster3D**	**1.41**	**412.45**	–	–	58.22	**0.0309**	**94.63**	0.4310	0.3804
Step1X-3D									
Baseline	1.00	704.76	16.70	0.143	46.40	0.0454	89.54	0.4401	0.3873
w/ **Faster3D**	**1.30**	**536.19**	16.57	0.144	**46.40**	0.0455	**89.70**	0.4394	**0.3907**

4.2 Main Results

Comparison with training-free acceleration. Table 1 compares **Faster3D** with full-compute TRELLIS and Fast3Dcache. **Faster3D** achieves a 1.342× Stage-I speedup, comparable to the 1.338× speedup of Fast3Dcache. At this comparable acceleration level, **Faster3D** better preserves both geometry and appearance. It achieves higher vIoU, indicating better occupancy agreement with ground-truth geometry, while higher PSNR and lower LPIPS demonstrate improved appearance quality. Across these metrics, its results remain close to those of full-compute TRELLIS. Meanwhile, **Faster3D** substantially reduces Stage-I Transformer FLOPs relative to full computation, yielding savings in both runtime and computational cost. Together, the quantitative results in Table 1 and qualitative comparisons in Figure 7 show that **Faster3D** better preserves geometry and appearance than Fast3Dcache at comparable acceleration.

Table 2: α **ablation on Toys4K-210.** Only α varies, with *CCR* fixed. Shaded rows denote the default configuration.

Setting	*Speed* ↑	LPIPS ↓	vIoU ↑	CD ↓
TRELLIS	1.000	0.0959	61.03	0.02785
$\alpha = 1$	1.317	0.0967	60.57	0.02788
$\alpha = 3$	1.342	0.0966	60.65	0.02777
$\alpha = 5$	1.385	0.0967	60.24	0.02779

Table 3: **Context ablations on Toys4K-210.** † denotes *TTD+CCR* without periodic refresh, retaining boundary evaluations.

Setting	*Speed* ↑	LPIPS ↓	vIoU ↑	CD ↓
TRELLIS	1.000	0.0959	61.03	0.02785
TTD-only	1.466	0.0993	58.17	0.03014
TTD+CCR†	1.416	0.0980	60.27	0.02878
Faster3D	1.342	0.0966	60.65	0.02777

Transfer to other 3D backbones. **Faster3D** transfers to TRELLIS.2, Hunyuan3D-2.1, TripoSG (Li et al., 2025c), and Step1X-3D, with separate calibration for each backbone. As shown in Table 1, it consistently accelerates structure generation across these backbones, with speedups ranging from $1.304\times$ to $1.954\times$. These runtime gains are accompanied by reduced Transformer FLOPs across all four backbones. Geometry and semantic alignment scores remain close to those of the corresponding full-compute baselines, suggesting that the acceleration largely preserves generation quality. Appearance metrics also show only small changes for backbones with appearance generation; TripoSG produces geometry only. Despite differences in latent representations and generation pipelines, all tested backbones benefit from velocity reutilization with preserved attention context. These results support the applicability of **Faster3D** beyond TRELLIS.

4.3 Ablation study

All ablations use TRELLIS on 210 assets with seed 42, 25 sampling steps, CFG scale 5, and unchanged Stage II. Activity ablations fix the fitted boundary, warmup, and CFG-off schedule. Tables 2 and 3 use interleaved full-compute and **Faster3D** reference runs with one warmup and three CUDA-synchronized repetitions. Speedups are averaged over paired per-asset ratios of median latencies.

Expansion factor. With *CCR* fixed, we vary $\alpha \in \{1, 3, 5\}$ in *TTD*. On 210 assets of Toys4K, Stage-I speedup increases from $1.317\times$ to $1.385\times$, with modest quality changes (Table 2). We retain $\alpha = 3$ because further acceleration is small. Appendix C.1.2 provides token-selection visualizations and the calibration-set speed–fidelity analysis (Figure 9).

Context restoration and refresh. With $\alpha = 3$ fixed, Table 3 evaluates the roles of CCR and cache refresh. *TTD-only* reutilizes token velocities without restoring cached K/V, while retaining the normal refresh schedule. *TTD+CCR* restores cached K/V but disables intermediate global refreshes; cache entries for freshly computed tokens are still updated. All-token Euler updates, cache initialization, boundary evaluations, and the CFG-off schedule remain unchanged. Removing cached K/V degrades geometry and appearance. Omitting intermediate refreshes affects geometry less but increases LPIPS. Full **Faster3D** performs best on all three quality metrics among accelerated variants, but with lower speedup. These ablations support context preservation and refresh, but cannot isolate K/V freshness because refreshes also update velocities.

With reused velocities fixed, retaining cached K/V reduces prediction error on active tokens and improves terminal occupancy agreement (Appendix C.3). K/V restoration also improves Fast3Dcache reconstruction quality at additional computational cost (Appendix C.4).

5 Conclusion

In this paper, we presented **Faster3D** as a training-free acceleration framework for image-to-3D generation. Our initial analysis reveals distinct roles of latent tokens during geometry generation. Many non-surface tokens exhibit stable velocities that allow velocity reutilization, yet they still provide important self-attention context for the remaining active tokens. Based on these observations, the first component of our method, *Temporal Token Decoupling*, uses activity from the latent prediction to select tokens for velocity reutilization. Then *Cached Context Restoration* restores cached keys and values for selected tokens at their original positions to preserve full self-attention context. By this means, our **Faster3D** reduces redundant computation without modifying pretrained weights. Experiments across diverse image-to-3D backbones demonstrate consistent acceleration of **Faster3D** while largely preserving the geometry and appearance quality of full-compute models.

AI USE STATEMENT

We used generative AI tools to assist with literature searches, suggest revisions to manuscript text, and check citation formatting and LaTeX layout. We did not use generative AI to develop the research method, implement experimental code, design experiments, generate or process datasets, or interpret experimental results. The authors reviewed the AI-assisted revisions and checked the suggested references against the original publications. The authors take responsibility for the accuracy, originality, and integrity of the final manuscript.

REFERENCES

Harry G. Barrow, Jay M. Tenenbaum, Robert C. Bolles, and Helen C. Wolf. Parametric correspondence and chamfer matching: Two new techniques for image matching. In *Proceedings of the Fifth International Joint Conference on Artificial Intelligence (IJCAI)*, pp. 659–663, 1977.

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 Fortieth AAAI Conference on Artificial Intelligence and Thirty-Eighth Conference on Innovative Applications of Artificial Intelligence and Sixteenth Symposium on Educational Advances in Artificial Intelligence*, AAAI'26/IAAI'26/EAAI'26. AAAI Press, 2026. ISBN 978-1-57735-906-7. doi: 10.1609/aaai.v40i24.39069. URL https://doi.org/10.1609/aaai.v40i24.39069.

Christopher B. Choy, Danfei Xu, JunYoung Gwak, Kevin Chen, and Silvio Savarese. 3d-r2n2: A unified approach for single and multi-view 3d object reconstruction. In Bastian Leibe, Jiri Matas, Nicu Sebe, and Max Welling (eds.), *Computer Vision – ECCV 2016*, pp. 628–644, Cham, 2016. Springer International Publishing. ISBN 978-3-319-46484-8.

Weilun Feng, Mingqiang Wu, zhiliang chen, Chuanguang Yang, Haotong Qin, Yuqi Li, Xiaokun Liu, Guoxin Fan, Libo Huang, Yulun Zhang, Michele Magno, Yongjun Xu, and Zhulin An. Fast-SAM3d: 3dfy anything in images but faster. In *Forty-third International Conference on Machine Learning*, 2026. URL https://openreview.net/forum?id=2kKk6W6oOF.

Jian Han, Jinlai Liu, Yi Jiang, Bin Yan, Yuqi Zhang, Zehuan Yuan, Bingyue Peng, and Xiaobing Liu. Infinity∞: Scaling bitwise autoregressive modeling for high-resolution image synthesis. In *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 15733–15744, 2025. doi: 10.1109/CVPR52734.2025.01467.

Danny Hernandez and Tom B. Brown. Measuring the algorithmic efficiency of neural networks, 2020. URL https://arxiv.org/abs/2005.04305.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin (eds.), *Advances in Neural Information Processing Systems*, volume 33, pp. 6840–6851. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper_files/paper/2020/file/4c5bcfec8584af0d967f1ab10179ca4b-Paper.pdf.

Team Hunyuan3D, Shuhui Yang, Mingxin Yang, Yifei Feng, Xin Huang, Sheng Zhang, Zebin He, Di Luo, Haolin Liu, Yunfei Zhao, Qingxiang Lin, Zeqiang Lai, Xianghui Yang, Huiwen Shi, Zibo Zhao, Bowen Zhang, Hongyu Yan, Lifu Wang, Sicong Liu, Jihong Zhang, Meng Chen, Liang Dong, Yiwen Jia, Yulin Cai, Jiaao Yu, Yixuan Tang, Dongyuan Guo, Junlin Yu, Hao Zhang, Zheng Ye, Peng He, Runzhou Wu, Shida Wei, Chao Zhang, Yonghao Tan, Yifu Sun, Lin Niu, Shirui Huang, Bojian Zheng, Shu Liu, Shilin Chen, Xiang Yuan, Xiaofeng Yang, Kai Liu, Jianchen Zhu, Peng Chen, Tian Liu, Di Wang, Yuhong Liu, Linus, Jie Jiang, Jingwei Huang, and Chunchao Guo. Hunyuan3d 2.1: From images to high-fidelity 3d assets with production-ready pbr material, 2025. URL https://arxiv.org/abs/2506.15442.

Kumara Kahatapitiya, Haozhe Liu, Sen He, Ding Liu, Menglin Jia, Chenyang Zhang, Michael S. Ryoo, and Tian Xie. Adaptive caching for faster video generation with diffusion transformers. In *2025 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 15240–15252, 2025. doi: 10.1109/ICCV51701.2025.01414.

Zeqiang Lai, Yunfei Zhao, Zibo Zhao, Haolin Liu, Fuyun Wang, Huiwen Shi, Xianghui Yang, Qingxiang Lin, Jingwei Huang, Yuhong Liu, Jie Jiang, Chunchao Guo, and Xiangyu Yue. Unleashing vecset diffusion model for fast shape generation. In *2025 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 2523–2533, 2025. doi: 10.1109/ICCV51701.2025.00243.

Jaeah Lee, Hyunjin Kim, Jaewoong Cho, and Gihyun Kwon. Vitality-aware compression for efficient image-to-shape diffusion transformers, 2026. URL https://arxiv.org/abs/2607.00382.

Kunjun Li, Zigeng Chen, Cheng-Yen Yang, and Jenq-Neng Hwang. Memory-efficient visual autoregressive modeling with scale-aware kv cache compression. In D. Belgrave, C. Zhang, H. Lin, R. Pascanu, P. Koniusz, M. Ghassemi, and N. Chen (eds.), *Advances in Neural Information Processing Systems*, volume 38, Main Conference, pp. 13736–13761. Curran Associates, Inc., 2025a. doi: 10.52202/085713-0462. URL https://proceedings.neurips.cc/paper_files/paper/2025/file/14550b451b65682b09f9080026d00a7e-Paper-Conference.pdf.

Senmao Li, Taihang Hu, Joost van de Weijer, Fahad Shahbaz Khan, Tao Liu, Linxuan Li, Shiqi Yang, Yaxing Wang, Ming-Ming Cheng, and Jian Yang. Faster diffusion: Rethinking the role of the encoder for diffusion model inference. In A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang (eds.), *Advances in Neural Information Processing Systems*, volume 37, pp. 85203–85240. Curran Associates, Inc., 2024. doi: 10.52202/079017-2706. URL https://proceedings.neurips.cc/paper_files/paper/2024/file/9ad996b5c45130de2bc00b60d8607904-Paper-Conference.pdf.

Senmao Li, Kai Wang, Salman Khan, Fahad Shahbaz Khan, jian Yang, and Yaxing Wang. FasterVAR: Plug-and-play acceleration for visual autoregressive models. In *Forty-third International Conference on Machine Learning*, 2026. URL https://openreview.net/forum?id=C6kxImRizt.

Weiyu Li, Xuanyang Zhang, Zheng Sun, Di Qi, Hao Li, Wei Cheng, Weiwei Cai, Shihao Wu, Jiarui Liu, Zihao Wang, Xiao Chen, Feipeng Tian, Jianxiong Pan, Zeming Li, Gang Yu, Xiangyu Zhang, Daxin Jiang, and Ping Tan. Step1x-3d: Towards high-fidelity and controllable generation of textured 3d assets, 2025b. URL https://arxiv.org/abs/2505.07747.

Yangguang Li, Zi-Xin Zou, Zexiang Liu, Dehu Wang, Yuan Liang, Zhipeng Yu, Xingchao Liu, Yuan-Chen Guo, Ding Liang, Wanli Ouyang, and Yan-Pei Cao. Triposg: High-fidelity 3d shape synthesis using large-scale rectified flow models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pp. 1–18, 2025c. doi: 10.1109/TPAMI.2025.3633512.

Yaron Lipman, Ricky T. Q. Chen, Heli Ben-Hamu, Maximilian Nickel, and Matthew Le. Flow matching for generative modeling. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=PqvMRDCJT9t.

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. In *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 7353–7363, 2025a. doi: 10.1109/CVPR52734.2025.00689.

Jiacheng Liu, Chang Zou, Yuanhuiyi Lyu, Junjie Chen, and Linfeng Zhang. From reusing to forecasting: Accelerating diffusion models with taylorseers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 15853–15863, October 2025b.

Kendong Liu, Zhiyu Zhu, Hui Liu, and Junhui Hou. Acc3d: Accelerating single image to 3d diffusion models via edge consistency guided score distillation. In *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 18031–18040, 2025c. doi: 10.1109/CVPR52734.2025.01680.

Xingchao Liu, Chengyue Gong, and Qiang Liu. Flow straight and fast: Learning to generate and transfer data with Rectified Flow. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=XVjTT1nw5z.

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 Y. Yue, A. Garg, N. Peng, F. Sha, and R. Yu (eds.), *International Conference on Learning Representations*, volume 2025, pp. 33132–33156, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/file/518046d86bbc41a0707727c38301ad8e-Paper-Conference.pdf.

Xinyin Ma, Gongfan Fang, and Xinchao Wang. Deepcache: Accelerating diffusion models for free. In *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 15762–15772, 2024. doi: 10.1109/CVPR52733.2024.01492.

William Peebles and Saining Xie. Scalable diffusion models with transformers. In *2023 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 4172–4182, 2023. doi: 10.1109/ICCV51070.2023.00387.

Albert Pumarola, Stefan Popov, Francesc Moreno-Noguer, and Vittorio Ferrari. C-flow: Conditional generative flow models for images and 3d point clouds. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

Ziran Qin, Youru Lv, Mingbao Lin, Hang Guo, Zeren Zhang, Danping Zou, and Weiyao Lin. Head-aware kv cache compression for efficient visual autoregressive modeling. In *Proceedings of the Fortieth AAAI Conference on Artificial Intelligence and Thirty-Eighth Conference on Innovative Applications of Artificial Intelligence and Sixteenth Symposium on Educational Advances in Artificial Intelligence*, AAAI'26/IAAI'26/EAAI'26. AAAI Press, 2026. ISBN 978-1-57735-906-7. doi: 10.1609/aaai.v40i30.39686. URL https://doi.org/10.1609/aaai.v40i30.39686.

Pratheba Selvaraju, Tianyu Ding, Tianyi Chen, Ilya Zharkov, and Luming Liang. Fora: Fast-forward caching in diffusion transformer acceleration, 2024. URL https://arxiv.org/abs/2407.01425.

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*, pp. 1798–1808, June 2021.

Maxim Tatarchenko, Stephan R. Richter, Rene Ranftl, Zhuwen Li, Vladlen Koltun, and Thomas Brox. What Do Single-View 3D Reconstruction Networks Learn? In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 352. IEEE, June 2019. doi: 10.1109/CVPR.2019.00352.

Keyu Tian, Yi Jiang, Zehuan Yuan, Bingyue Peng, and Liwei Wang. Visual autoregressive modeling: Scalable image generation via next-scale prediction. In A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang (eds.), *Advances in Neural Information Processing Systems*, volume 37, pp. 84839–84865. Curran Associates, Inc., 2024. doi: 10.52202/079017-2694. URL https://proceedings.neurips.cc/paper_files/paper/2024/file/9a24e284b187f662681440ba15c416fb-Paper-Conference.pdf.

Jianfeng Xiang, Zelong Lv, Sicheng Xu, Yu Deng, Ruicheng Wang, Bowen Zhang, Dong Chen, Xin Tong, and Jiaolong Yang. Structured 3d latents for scalable and versatile 3d generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 21469–21480, June 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.

Guangxuan Xiao, Yuandong Tian, Beidi Chen, Song Han, and Mike Lewis. Efficient streaming language models with attention sinks. In B. Kim, Y. Yue, S. Chaudhuri, K. Fragkiadaki, M. Khan, and Y. Sun (eds.), *International Conference on Learning Representations*, volume 2024, pp. 21875–21895, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/file/5e5fd18f863cbe6d8ae392a93fd271c9-Paper-Conference.pdf.

Le Xue, Ning Yu, Shu Zhang, Artemis Panagopoulou, Junnan Li, Roberto Martín-Martín, Jiajun Wu, Caiming Xiong, Ran Xu, Juan Carlos Niebles, and Silvio Savarese. Ulip-2: Towards scalable multimodal pre-training for 3d understanding. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 27091–27101, June 2024.

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 (CVPR)*, pp. 27030–27040, June 2026.

Xingyi Yang, Songhua Liu, and Xinchao Wang. Hash3d: Training-free acceleration for 3d generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 21481–21491, June 2025.

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*, pp. 586–595, 2018.

Zhenyu Zhang, Ying Sheng, Tianyi Zhou, Tianlong Chen, Lianmin Zheng, Ruisi Cai, Zhao Song, Yuandong Tian, Christopher Ré, Clark Barrett, Zhangyang "Atlas" Wang, and Beidi Chen. H2o: Heavy-hitter oracle for efficient generative inference of large language models. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine (eds.), *Advances in Neural Information Processing Systems*, volume 36, pp. 34661–34710. Curran Associates, Inc., 2023. doi: 10.52202/075280-1506. URL https://proceedings.neurips.cc/paper_files/paper/2023/file/6ceefa7b15572587b78ecfcebb2827f8-Paper-Conference.pdf.

Xuanlei Zhao, Xiaolong Jin, Kai Wang, and Yang You. Real-time video generation with pyramid attention broadcast. In Y. Yue, A. Garg, N. Peng, F. Sha, and R. Yu (eds.), *International Conference on Learning Representations*, volume 2025, pp. 3296–3319, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/file/092c2d45005ea2db40fc24c470663416-Paper-Conference.pdf.

Junsheng Zhou, Jinsheng Wang, Baorui Ma, Yu-Shen Liu, Tiejun Huang, and Xinlong Wang. Uni3d: Exploring unified 3d representation at scale. In B. Kim, Y. Yue, S. Chaudhuri, K. Fragkiadaki, M. Khan, and Y. Sun (eds.), *International Conference on Learning Representations*, volume 2024, pp. 46766–46782, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/file/cc2c03ce028a1d6cbed6f98cede7f5a9-Paper-Conference.pdf.

Zanwei Zhou, Taoran Yi, Jiemin Fang, Chen Yang, Lingxi Xie, Xinggang Wang, Wei Shen, and Qi Tian. Few-step flow for 3d generation via marginal-data transport distillation. In *Proceedings of the Fortieth AAAI Conference on Artificial Intelligence and Thirty-Eighth Conference on Innovative Applications of Artificial Intelligence and Sixteenth Symposium on Educational Advances in Artificial Intelligence*, AAAI'26/IAAI'26/EAAI'26. AAAI Press, 2026. ISBN 978-1-57735-906-7. doi: 10.1609/aaai.v40i16.38394. URL https://doi.org/10.1609/aaai.v40i16.38394.

Chang Zou, Xuyang Liu, Ting Liu, Siteng Huang, and Linfeng Zhang. Accelerating diffusion transformers with token-wise feature caching. In Y. Yue, A. Garg, N. Peng, F. Sha, and R. Yu (eds.), *International Conference on Learning Representations*, volume 2025, pp. 75467–75488, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/file/bbe024e0517fe12ac3a8d388b19ff9fe-Paper-Conference.pdf.

Appendix Contents

Appendix A Statements . . . 14
A.1 Limitations . . . 14
A.2 Ethical Statement . . . 15
A.3 Reproducibility Statement . . . 15
Appendix B Experimental Details . . . 15
B.1 Datasets and Conditioning Images . . . 15
B.2 Generation and Efficiency Measurement . . . 16
B.3 Transfer Settings and Ablation Scope . . . 17
B.4 Geometry Metrics . . . 18
B.5 Rendered Appearance Metrics . . . 19
B.6 Image-3D Semantic Consistency . . . 20
Appendix C Additional Method Details and Analysis . . . 21
C.1 Temporal Token Decoupling (TTD) . . . 21
C.2 Cached Context Restoration (CCR) . . . 25
C.3 Controlled K/V Context Interventions . . . 27
C.4 Cross-Method Cached Context Restoration . . . 28
C.5 Token Selection at Comparable Runtime . . . 29
Appendix D Additional Qualitative Results . . . 30

A Statements

A.1 Limitations.

Faster3D accelerates flow-based image-to-3D generation, but its applicability to other generative paradigms remains unverified. *Temporal Token Decoupling (TTD)* uses activity from the latent prediction to guide velocity reutilization. This activity is an empirical proxy, not an exact indicator of surface membership or prediction error. The reported transfer experiments use configurations calibrated offline and fixed before evaluation. Transfer to further backbones may require calibration, and more aggressive velocity reutilization can reduce generation quality. *Cached Context Restoration (CCR)* preserves the full self-attention context, but cached keys and values may differ from freshly computed ones. Periodic refreshes limit cache staleness without guaranteeing equivalence to full computation. Maintaining K/V caches also requires memory. Finally, our reported stage-wise speedups do not directly translate into equal end-to-end gains, since downstream generation and decoding remain unchanged.

A.2 Ethical Statement.

Faster3D operates on existing pretrained backbones without updating their weights. It does not address biases, privacy concerns, or content-safety limitations inherited from these models. Deployment should respect the licenses and usage conditions of the underlying models and datasets, as well as the rights associated with input images and generated assets. Our acceleration method should not be interpreted as providing additional safeguards against misuse.

A.3 Reproducibility Statement.

The main text and appendices describe the *TTD* and *CCR* mechanisms, calibration procedure, sampling configurations, and evaluation protocols. Calibration data are separate from the test set, and calibration does not update model weights. Table 5 reports the complete inference and calibration configuration for each backbone, including calibration data source and size, separation from the evaluation set, latent representation, accelerated stage, activity definition, threshold parameters, warmup length, admission observations, refresh parameters, CFG transition, late-stage policy, and the Fast3Dcache adaptation and tuning procedure. For backbones without grid-aligned latent tokens, the table also specifies the criterion used to calibrate the activity boundary.

B Experimental Details

B.1 Datasets and Conditioning Images

Toys4K-210. Toys4K (Stojanov et al., 2021) contains over 4,000 textured 3D objects from 105 categories. We construct a category-balanced subset with two objects per category, giving 210 objects in total. Within each category, we sort the available Blender asset paths lexicographically and select the first two. The resulting asset identifiers and rendering settings are stored in a fixed manifest. Selection is therefore reproducible and does not depend on the outputs of any generation method. All methods receive the same conditioning image for each object and are evaluated against the same ground-truth asset. Each object contributes one input image, not multiple independently evaluated prompts.

Calibration and analysis. We calibrate activity boundaries and operating points offline on calibration sets disjoint from Toys4K-210, without updating model weights. The TRELLIS (Xiang et al., 2025) calibration set comprises 42 example images. Transfer configurations use calibration images from the corresponding backbone and are fixed before testing on Toys4K-210, with no further parameter search on the test set. Toys4K-210 evaluates reconstruction against ground-truth assets. Calibration analyses instead measure agreement with full-compute outputs at the same seed, which is distinct from ground-truth vIoU.

Object normalization and input camera. We preserve the source materials and normalize each object using its world-space mesh bounding box. Let $\mathbf{b}_{\min}$ and $\mathbf{b}_{\max}$ be its bounds. A vertex $\mathbf{x}$ is transformed as:

$$\mathbf{x}' = \frac{\mathbf{x} - (\mathbf{b}_{\min} + \mathbf{b}_{\max})/2}{\max_j (b_{\max,j} - b_{\min,j})}. \tag{4}$$

The object is centered at the origin and its longest extent is one. Input images use a perspective camera at radius 2, azimuth 210°, and elevation 25°, with a 40° field of view. In Blender's z-up coordinates, the camera position is $2(\cos\theta\cos\phi, \sin\theta\cos\phi, \sin\phi)$ and it points toward the origin. This camera is fixed across objects and methods; no method-specific view selection is used.

Input rendering. We render 1024×1024 RGBA images with Blender Cycles, 128 samples, denoising, and a transparent film. The lighting setup follows the released TRELLIS (Xiang et al., 2025) rendering script: a point light at $(4, 1, 6)$ with energy 1,000, an upper area light at $(0, 0, 10)$ with energy 10,000 and object scale $(100, 100, 100)$, and a lower area light at $(0, 0, -10)$ with energy 1,000. The input scene has no world illumination. Exposure is zero and gamma is one. The renderer uses Filmic when available and AgX otherwise. We crop to the alpha foreground, add 5% padding on each side, form a square canvas, and resize it to 1024×1024. The input-render seed

Table 4: Sampling and timing settings. Warmups are untimed benchmark runs, not the initial full-compute steps within **Faster3D**. TRELLIS warms up each asset–method pair; the other runners warm up each method before evaluating the dataset. Repetitions are per asset and method.

Backbone	Sampling steps	Guidance	Warmups	Repetitions
TRELLIS	25	5.0	1	3
TRELLIS.2	12	7.5	1	1
Hunyuan3D-2.1	50	5.0	3	3
TripoSG	50	7.0	5	1
Step1X-3D	50	7.5	3	3

is fixed to 20260817. This one-time rendering seed is distinct from the generation and metric seed, which is 42. Each backbone then applies its native image preprocessing to the same saved input.

B.2 Generation and Efficiency Measurement

Paired generation. We use the same pretrained weights, conditioning image, sampling schedule, guidance settings, and seed for the full-compute and accelerated versions of each backbone. The main comparison changes only the targeted flow-transformer computation. For TRELLIS, Stage I denotes sparse-structure generation and Stage II denotes structured latent (SLAT) generation. Faster3D modifies Stage I and preserves Stage II and asset decoding. The downstream stages of each transfer backbone also remain unchanged. Table 4 lists the sampling settings and timing repetitions. TRELLIS uses 25 Euler steps with guidance scale 5 and time rescaling factor 3. TRELLIS.2 uses its released sparse-structure configuration: 12 steps, guidance strength 7.5, guidance rescale 0.7, guidance interval $[0.6, 1.0]$, and time rescaling factor 5. Guidance values are held fixed within each comparison; they need not be equal across different backbones.

Wall-clock timing. Each inference run uses one NVIDIA RTX PRO 6000 Blackwell GPU with batch size one. Models remain resident during timing. CUDA synchronization brackets the measured sampling interval. Model loading, image encoding, mesh decoding, texture generation, file export, and metric computation are outside this interval. *TTD* selection, *CCR*, and cache-refresh overhead inside the sampler are included. The reported stage is the sparse-structure sampler for TRELLIS and TRELLIS.2, and the shape/geometry sampler for Hunyuan3D-2.1, TripoSG, and Step1X-3D. These measurements are not end-to-end textured-asset speedups.

For repeated measurements, we use the median latency of each asset–method pair. Let $T_{i,m}$ denote this latency and $T_{i,\text{full}}$ its full-compute reference. The paired speedup is:

$$S_m = \frac{1}{M} \sum_{i=1}^{M} \frac{T_{i,\text{full}}}{T_{i,m}}. \tag{5}$$

The transfer runners alternate method order across assets, and across repetitions when present. The TRELLIS ablations use interleaved full-compute, **Faster3D**, and ablation runs with matched noise tensors. Their timing batches are separate from the main comparison. Thus, a reference speedup can differ slightly between tables even when its generation settings are unchanged. The single-repetition TRELLIS.2 and TripoSG runs provide paired dataset-level timings, but do not measure within-asset timing variance.

Model-core FLOPs (Hernandez & Brown, 2020). We report analytical operation counts in TFLOPs, where one TFLOP is 10^{12} floating-point operations and a multiply–add counts as two operations. The counter combines the model dimensions with the recorded number of executed branches, blocks, fresh queries, and context tokens at each sampling step. For example, the two attention matrix products require $4qkd$ operations for q queries, k context tokens, and hidden width d. *TTD* reduces q, while *CCR* retains the full k in self-attention. Whole-step velocity reutilization executes no Transformer blocks. Conditional and unconditional evaluations are counted separately when CFG is active.

For TRELLIS, we retain the operation-count conventions of the released Fast3Dcache counter and distinguish fresh-query rows from context rows. Backbone-specific counters account for the corre-

sponding Transformer architecture in the transfer experiments. Counts include the modeled attention, projections, MLPs, normalization, and modulation operations, but exclude image encoders, input/output adapters outside the counted core, decoders, rendering, and export. These are model-core estimates, not hardware-profiler measurements or estimates of memory traffic. We report wall-clock speedup separately because cache movement and kernel overhead are not captured by FLOPs alone.

B.3 Transfer Settings and Ablation Scope

Transfer settings. The transfer implementations use fixed activity boundaries, warmup lengths, admission settings, and maximum refresh intervals established before testing. Two new observations below the activity threshold are required before admission in the four transfer configurations, whereas the main TRELLIS configuration requires one. The shared components are *TTD* for selecting tokens for velocity reutilization and *CCR* for restoring their K/V context, with periodic cache refresh. The spatial surface and non-surface labels defined for the TRELLIS grid are used only for offline analysis and calibration, not as runtime labels or as a geometric interpretation of latent slots in every backbone.

Table 5: Calibration and acceleration settings for Table 1, supplementing the sampling and timing settings in Table 4. Step indices start at zero.

(a) Calibration and velocity reutilization

Setting	TRELLIS	TRELLIS.2	Hunyuan3D 2.1	TripoSG	Step1X-3D
Latent representation	Grid	Grid	Slots	VectSet	Slots
Tokens × channels	$16^3 \times 8$	$16^3 \times 8$	4096×64	2048×64	2048×64
Calibration images	42	72	108	48	117
Calibration seed	42	42	42	42	42
Activity percentile	–	–	80th	30th	50th
Expansion α	3	1.5	1	1.1	3
Initial full steps	0–5	0–2	0–3	0–9	0–23
First activity step	6	0	2	10	24
Cache initialization	6	3	4	10	24
Refresh $(K_{\min}, K, \delta)$	$(2, 5, 1.21)$	$(2, 4, 0.75)$	$(2, 6, 1.0)$	$(2, 4, 0.75)$	Fixed $K = 2$

(b) Calibration criteria and boundary settings

Definitions	Activity and admission follow Appendix C.1; refresh parameters follow Appendix C.2. For the percentile settings in (a), we compute the specified percentile of token activity within each calibration example at each step, then take the median across examples to construct the boundary. These settings do not require spatial labels. Boundary ranges below are before multiplication by α. Calibration quality is measured against the corresponding full-compute reference.
TRELLIS	Calibration uses project example images. The spatial labels, admission criterion, and logistic coefficients are given in Appendix C.1. After CFG is disabled, the complete velocity field is reutilized according to Appendix C.1.1.
TRELLIS.2	Calibration uses released example images. A subset of 24 selects the fastest tested scale with mean occupancy agreement ≥ 0.90. The retained labels use decoded zero crossings or the exterior topology boundary, without dilation, with the same admission criterion as TRELLIS. The logistic fit over steps 3–9 uses $(a, b, k, t_0) = (0.687638, 1.179639, 1.354410, 7.938406)$, including the downward shift 0.022810. Steps 0–2 use a boundary of 0.61. After CFG is disabled, steps 10 and 11 use full conditional computation.
Hunyuan3D-2.1	Calibration uses released example images; the final density sweep uses eight priority examples. The percentile boundary uses second-difference smoothing with weight 4 and ranges from 8.542635 to 9.158007.

Continued on the next page

Table 5: Calibration and acceleration settings (continued).

TripoSG	Calibration uses released example images. The percentile boundary uses second-difference smoothing with weight 4. Configuration selection requires mean F-score ≥ 0.95, 10th-percentile F-score ≥ 0.90, and mean relative terminal latent error ≤ 0.30. The F-score distance threshold is 0.02. The boundary ranges from 6.964913 to 7.669686; steps 31–49 retain the fitted value at step 30.
Step1X-3D	Calibration uses released example images. Candidate configurations vary the activity percentile, threshold scale, and initial full computation interval. The selected percentile boundary ranges from 7.257139 to 7.934547. Refreshes use the fixed interval in (a), without an accumulated-score test.
CFG policy	Hunyuan3D-2.1, TripoSG, and Step1X-3D retain CFG throughout sampling and continue the same token selection and refresh policy, with no separate final phase.
(c) Fast3Dcache adaptation and tuning	
TRELLIS	The existing implementation is connected through a sampler adapter without refitting PCSC. Settings are: PCSC slope -0.07, anchor 5, initial full computation setting 5, SSC threshold 3, acceleration weight 0.7, correction frequency 3, final-phase boundary 19, and final cache ratio 0.7. TRELLIS guidance strength 5 maps to Fast3Dcache guidance scale 6.
TRELLIS.2	The adaptation uses native grid tokens and 64^3 decoder probes. Of 72 released examples, 24 are used for calibration and 48 reserved for validation. Tuning searches initial full computation settings $\{2, 3, 4\}$, SSC thresholds $\{2, 3, 4\}$, and final cache ratios $\{0, 0.7\}$. The reported configuration uses PCSC slope 0.136600, anchor 1, initial setting 4, SSC threshold 2, acceleration weight 0.7, correction frequency 3, and final cache ratio 0. It was selected manually before evaluation after the 0.95 and 0.90 occupancy-agreement gates failed. The reserved validation was not run.
Hunyuan3D-2.1	The adaptation uses 36 released examples for calibration and 72 for validation. Two online 64^3 decoder probes determine a global active-slot budget; SSC selects slot identities without assuming voxel correspondence. Tuning searches initial full computation settings $\{16, 20, 24, 36, 44\}$ and SSC thresholds $\{2, 3, 4\}$, with a mean occupancy-agreement requirement of 0.90. The selected configuration uses PCSC slope -0.192105, anchor 15, initial setting 16, SSC threshold 4, and acceleration weight 0.7.

Ablation scope. The main-text ablations test threshold expansion, context restoration, and refresh. Table 2 varies $\alpha \in \{1, 3, 5\}$ in *TTD* while keeping *CCR* fixed. Table 3 fixes $\alpha = 3$: *TTD-only* omits cached K/V restoration but retains normal refreshes, and *TTD+CCR*† restores cached K/V but omits intermediate global refreshes. Both retain velocity reutilization, Euler updates for all tokens, cache initialization, boundary evaluations, and the same CFG-off schedule (Appendix C.1.1). Refreshes update both velocities and K/V, so these ablations do not isolate K/V freshness.

B.4 Geometry Metrics

Surface extraction and alignment. We extract the generated mesh from its GLB and the ground-truth mesh from its Blender asset. Both are independently normalized by Eq. 4. To focus on externally visible geometry, we rasterize each mesh from 100 deterministic cameras distributed over the sphere by a centered Hammersley sequence. The cameras have radius 2, field of view $40°$, and resolution 512×512. We collect visible surface positions and uniformly subsample these rasterized samples to 100,000 points with metric seed 42. This is sampling from visible pixels, rather than uniform sampling over mesh area.

We align the predicted points to the ground truth by rigid point-to-point ICP, without an additional scale fit. ICP uses 20,000-point subsets and four initial yaw rotations: $0°$, $90°$, $180°$, and $270°$. Coarse and fine correspondence thresholds are 0.20 and 0.05, with at most 120 and 200 iterations. We choose the alignment with the lowest bidirectional mean nearest-neighbor distance on these subsets. The selected transform is applied to the complete predicted point set and mesh. This removes global pose ambiguity before computing geometry and paired appearance scores.

Chamfer Distance (Barrow et al., 1977; Pumarola et al., 2020). CD measures overall surface discrepancy. Let P and G be the aligned predicted and ground-truth point sets, and let $d(\mathbf{x}, Y) = \min_{\mathbf{y}\in Y} \|\mathbf{x} - \mathbf{y}\|_2$. We compute

$$\mathrm{CD}(P, G) = \frac{1}{|P|} \sum_{\mathbf{p}\in P} d(\mathbf{p}, G) + \frac{1}{|G|} \sum_{\mathbf{g}\in G} d(\mathbf{g}, P). \tag{6}$$

Our CD uses unsquared Euclidean distances and sums the two directions without a factor of $1/2$. Lower values are better. Values from squared-distance or differently normalized implementations are therefore not directly interchangeable with ours.

F-score (Tatarchenko et al., 2019). F-score complements CD by measuring the fraction of the two surfaces that match within a fixed tolerance. With $\eta = 0.05$ in the normalized coordinate system, we compute

$$\begin{aligned} \mathrm{Prec}_\eta &= \frac{1}{|P|} \sum_{\mathbf{p}\in P} \mathbb{I}[d(\mathbf{p}, G) \le \eta], \\ \mathrm{Rec}_\eta &= \frac{1}{|G|} \sum_{\mathbf{g}\in G} \mathbb{I}[d(\mathbf{g}, P) \le \eta], \\ F_\eta &= \frac{2\,\mathrm{Prec}_\eta\,\mathrm{Rec}_\eta}{\mathrm{Prec}_\eta + \mathrm{Rec}_\eta}. \end{aligned} \tag{7}$$

We set $F_\eta = 0$ if both precision and recall are zero, and report $100F_\eta$. This threshold is 5% of the normalized object's longest extent. Higher scores indicate better surface precision and coverage.

Volumetric IoU (Choy et al., 2016). vIoU measures agreement in occupied 3D support, including missing and excess regions. We voxelize the normalized, aligned meshes on a common 32^3 grid covering $[-0.5, 0.5]^3$. The implementation uses subdivided surface voxelization with pitch $1/32$, followed by binary hole filling for enclosed regions. It does not repair an open mesh into a watertight solid. For occupied voxel sets V_P and V_G,

$$\mathrm{vIoU} = \frac{|V_P \cap V_G|}{|V_P \cup V_G|}. \tag{8}$$

We report $100\,\mathrm{vIoU}$, with higher values being better. This mesh-to-ground-truth score differs from the 64^3 decoder-occupancy agreement used in analyses on the TRELLIS calibration set.

B.5 Rendered Appearance Metrics

Paired evaluation views. Appearance evaluation uses one deterministic view per asset, separate from the conditioning-image camera. For manifest index $i \in \{0, \ldots, 209\}$, we set $u_i = (i + 1/2)/210$, $v_i = \phi_2(i)$, azimuth $\theta_i = 2\pi v_i$, and elevation $\varphi_i = \arccos(1 - 2u_i) - \pi/2$, where ϕ_2 is the base-two radical inverse. This distributes views over the sphere across the dataset. It is not a 210-view evaluation of each object. Prediction and ground truth use the same camera, radius 2, field of view $40°$, and 512×512 resolution, after the alignment described above.

Ground-truth images use the source materials and the same three-light layout as the input renderer, with Cycles, 128 samples, denoising, and metric seed 42. TRELLIS predictions are rendered from their saved native Gaussian representation; the GLB supplies geometry for alignment, not the appearance being scored. The Gaussian renderer uses an OpenCV-style look-at transform toward the origin (world $+z$ as up and camera $+z$ as forward, with positive camera-space depth), with camera radius 2.0 and near and far parameters 0.8 and 1.6, respectively. These settings are applied identically to the full-compute and accelerated outputs. The ICP alignment transform is applied to the Gaussian representation by composing it with the normalization transform and camera extrinsics, $\mathbf{E}_{\mathrm{G}} = \mathbf{E}_{\mathrm{aligned}}\mathbf{T}_{\mathrm{ICP}}\mathbf{T}_{\mathrm{norm}}$. The stored Gaussian centers, covariance parameters, and spherical harmonic coefficients remain unchanged, as alignment is handled through the composed camera transform. TRELLIS.2, Hunyuan3D-2.1, and Step1X-3D predictions use their exported textured meshes and the paired Blender rendering path. TripoSG produces geometry only in our experiments, so we do not report PSNR or LPIPS for it. Renderer differences across backbones are retained and

disclosed; comparisons between a backbone and its accelerated version use the same representation and renderer.

Before scoring, we composite both RGBA images onto white as $I = \alpha I_{\mathrm{RGB}} + (1-\alpha)\mathbf{1}$, with RGB values in $[0,1]$. Scores use the entire image. We do not crop each method's foreground independently or select its best-looking view.

PSNR. PSNR measures pixel-level appearance agreement, including color and silhouette differences. For paired images $I_P, I_G \in [0,1]^{H\times W\times 3}$,

$$\mathrm{MSE} = \frac{\|I_P - I_G\|_2^2}{3HW}, \qquad \mathrm{PSNR} = 10\log_{10}\frac{1}{\mathrm{MSE}}. \tag{9}$$

We report PSNR in decibels and average per-image scores, rather than computing PSNR from a dataset-averaged MSE. Higher values are better.

LPIPS (Zhang et al., 2018). LPIPS measures perceptual differences that are not fully captured by pixel error. We use the pretrained VGG-based LPIPS network in evaluation mode and map each RGB image from $[0,1]$ to $[-1,1]$. In the standard LPIPS formulation,

$$\mathrm{LPIPS}(I_P, I_G) = \sum_{\ell} \frac{1}{H_\ell W_\ell} \sum_{h,w} \left\| \mathbf{w}_\ell \odot \left(\widehat{\mathbf{f}}_\ell^P(h,w) - \widehat{\mathbf{f}}_\ell^G(h,w)\right)\right\|_2^2, \tag{10}$$

where $\widehat{\mathbf{f}}_\ell$ denotes channel-normalized features and $\mathbf{w}_\ell$ denotes the learned channel weighting. The network is not fitted to our data. Lower scores are better. Both PSNR and LPIPS are view-dependent appearance measures, not complete evaluations of material properties under arbitrary lighting.

B.6 Image-3D Semantic Consistency

Colored point clouds. ULIP-2 (Xue et al., 2024) and Uni3D (Zhou et al., 2024) measure whether a generated object remains semantically consistent with its conditioning image. They complement geometric reconstruction scores when more than one plausible 3D shape can explain an image. For geometry-only outputs, RGB channels are constructed using constant neutral-gray face colors, with $R = G = B = 0.4$ for every sampled point. Semantic comparisons use the same valid assets across methods within each backbone: 208 assets for TRELLIS and all 210 assets for TRELLIS.2, Hunyuan3D-2.1, TripoSG, and Step1X-3D. The evaluation manifest lists the excluded asset identifiers and exclusion reasons. Coordinates are centered by their point mean and scaled to unit maximum radius. We then apply deterministic farthest-point sampling to obtain 10,000 XYZRGB points. This normalization is separate from the bounding-box normalization and ICP used for geometry scores. The same sampled point cloud is reused by ULIP-2 and Uni3D, with metric seed 42 and fixed sampling settings across methods.

Embedding similarity. We use the released colored PointBERT checkpoint of ULIP-2 with its ViT-bigG-14 image encoder. For Uni3D, we use the released Uni3D-g checkpoint with the EVA02-E-14-plus image encoder. Each model applies its own image preprocessing to the original conditioning image, not to a selected rendering of the generated object. Let f_{img} and f_{3D} denote the corresponding encoders. The score is:

$$\mathrm{Sim}(I,P) = \frac{f_{\mathrm{img}}(I)^{\mathsf{T}} f_{\mathrm{3D}}(P)}{\|f_{\mathrm{img}}(I)\|_2 \, \|f_{\mathrm{3D}}(P)\|_2}. \tag{11}$$

We report the raw cosine similarity without a learned logit scale or percentage conversion. Higher values are better. These scores assess image–3D agreement, not similarity to a ground-truth 3D embedding.

Aggregation and valid samples. Quality scores are arithmetic means over evaluated assets. With two assets per category, the complete 210-object mean also weights categories equally. We retain the valid sample count for each metric and do not replace missing results with zeros. The main TRELLIS comparison has 210 evaluated assets for geometry and appearance, and 208 for ULIP-2 and Uni3D. The semantic scores of the Fast3Dcache adaptations use separate paired evaluations

and are excluded from ranking and heatmaps in Table 1. Sampling-based geometry metrics, point-cloud construction, and appearance rendering use the same base metric seed for a given asset across methods. Identical seeds control these stochastic procedures; they do not imply bitwise-identical outputs across different renderers or hardware configurations.

C Additional Method Details and Analysis

TTD selects tokens for velocity reutilization (Section 3.2), while *CCR* restores their cached K/V context (Section 3.3). Periodic full-token evaluations refresh both velocities and K/V. Unless stated otherwise, this section describes the main TRELLIS configuration. As in Sections 3.2 and 3.3, t denotes a zero-based sampling step. We write u_t for the corresponding continuous flow time to distinguish it from the step index. The symbol s remains the CFG guidance strength defined in Section 3.1.

C.1 Temporal Token Decoupling (TTD)

Activity computation. TRELLIS Stage I has $N = 4096$ tokens on a 16^3 grid, with $C = 8$ latent channels. Its Transformer has 24 blocks of width 1024 and 16 attention heads. At step t, the current latent and sampler velocity give the latent prediction through the native conversion,

$$\widehat{\mathbf{x}}_{0,t} = (1 - \sigma_{\min})\mathbf{x}_{u_t} - \sigma(u_t)\mathbf{v}_t, \qquad \sigma(u_t) = \sigma_{\min} + (1 - \sigma_{\min})u_t. \tag{12}$$

The sampler velocity $\mathbf{v}_t$ includes the CFG combination defined in Section 3.1. When $\sigma_{\min} = 0$, the expression reduces to $\mathbf{x}_{u_t} - u_t\mathbf{v}_t$. The latent prediction $\widehat{\mathbf{x}}_{0,t} \in \mathbb{R}^{C\times N}$ estimates the final latent representation, not a binary occupancy mask. We standardize each channel over all N tokens within the sample,

$$\widetilde{x}_{0,t}(c,i) = \frac{\widehat{\mathbf{x}}_{0,t}(c,i) - \mu_{t,c}}{s_{t,c} + \varepsilon}, \tag{13}$$

where $\mu_{t,c}$ and $s_{t,c}$ are the channel mean and population standard deviation, and $\varepsilon = 10^{-6}$, as in Eq. 2. The token activity score is $A_t(i) = \left[\sum_c \widetilde{x}_{0,t}(c,i)^2\right]^{1/2}$. Token activity is a proxy for velocity reutilization, not an exact measure of temporal stability or surface membership. Token selection requires no additional DiT evaluation or intermediate decoding.

Conservative boundary. We calibrate on the TRELLIS calibration set, separately from the test data. Let $L_j(v)$ be the final decoded occupancy logit at voxel v and B_i the spatially corresponding $4 \times 4 \times 4$ voxel block for token i. Following Section 3.2, i belongs to the surface-token set $\mathcal{S}_j$ if $\max_{v\in B_i} L_j(v) > 0$, and is non-surface otherwise. Thus, a block with all positive logits is also labeled surface. The block denotes spatial correspondence, not an exclusive decoder receptive field. These retrospective labels are used only for offline analysis and calibration. At threshold u, the fraction of surface tokens admitted is:

$$L_{j,t}(u) = \frac{1}{|\mathcal{S}_j|} \sum_{i\in\mathcal{S}_j} \mathbb{I}[A_{j,t}(i) \le u]. \tag{14}$$

We scan thresholds from 0.05 to 4.00 in increments of 0.01. At each step, we select the largest threshold whose cross-example 95th-percentile leakage is at most 0.005. This is an empirical surface-protection criterion, not a bound on generation error.

We fit a logistic curve over steps 6–18 and shift it downward until it lies below the empirical thresholds at these steps. The resulting runtime boundary is:

$$\widehat{\tau}_E(t) = a + \frac{b - a}{1 + \exp[-k(t - t_0)]}, \tag{15}$$

where a and b include the downward shift and t_0 is the inflection-step location. The resulting parameters are $a = 0.456044$, $b = 1.391186$, $k = 0.691227$, and $t_0 = 11.244633$. The downward shift is 0.067896. The implementation stores the full-precision schedule. It does not refit the curve during generation.

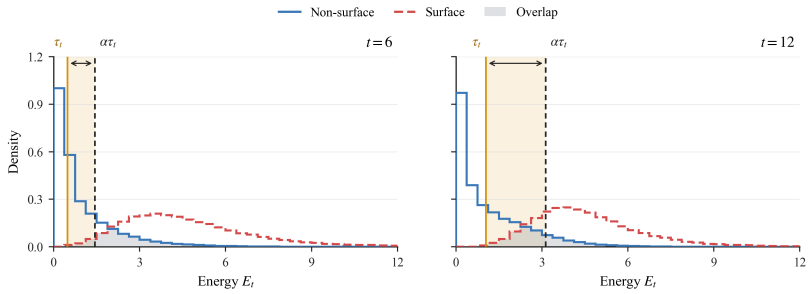


Figure 8: **Overlapping token activity distributions motivate *TTD*.** Histograms pool 42 calibration examples at $t = 6$ and $t = 12$, with each class normalized separately and final-surface labels used only retrospectively. In the plot, E_t denotes the activity score A_t, and τ_t denotes $\hat{\tau}_E(t)$. Gray marks distribution overlap; the gold band shows threshold expansion with $\alpha = 3$, which admits tokens from both classes. Densities use the full support; the displayed range omits at most 1.19% of either class.

Threshold expansion. *TTD* forms the candidate set $\mathcal{C}_t = \{i \mid A_t(i) \leq \alpha\widehat{\tau}_E(t)\}$. The default is $\alpha = 3$. Surface and non-surface tokens overlap in activity (Figure 8), so expanding the boundary admits low-activity tokens from both groups. Tokens above the threshold remain ineligible. The expanded set need not satisfy the original surface-leakage criterion: α controls the quality–efficiency trade-off, not surface-classification accuracy. Appendix C.1.2 examines the selected tokens and the resulting trade-off.

Causal admission. Steps 0–5 use full computation during the rapidly changing early phase. Step 6 initializes the caches and provides the first activity observation. A freshly evaluated token with finite activity at or below the expanded boundary becomes eligible at the next step. Velocity reutilization can therefore start at step 7. Eligibility is reassessed only after a new DiT prediction, including at global refreshes. A cached velocity is not treated as a new observation of token stability.

C.1.1 CFG Transition and Late-Stage Velocity Reutilization

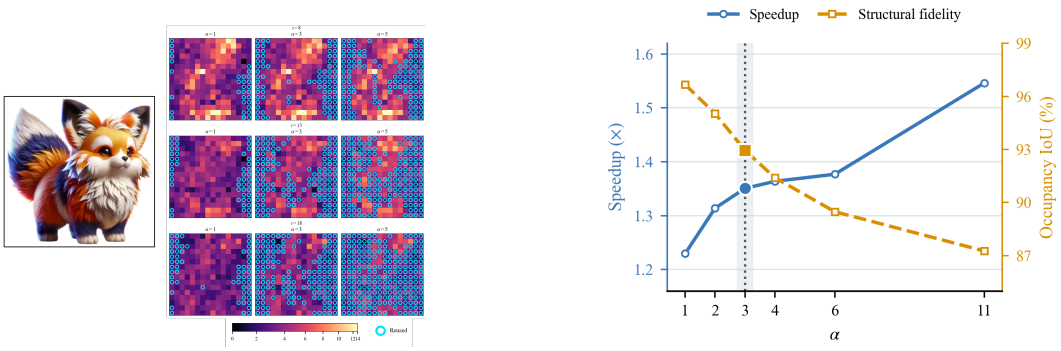
In an offline diagnostic after CFG is disabled, the activity boundary is nearly constant, with a conservative constant fit $\tau_{\text{off}} = 1.51$ examined at $\alpha = 5$. This diagnostic does not replace the default $\alpha = 3$ used for token selection while CFG is active. The velocity dynamics in Figure 4 also become more similar in this phase. The implemented late-stage policy therefore uses a fixed schedule for reutilizing the complete velocity field, rather than applying this diagnostic threshold during the final five steps.

We use a shorter, three-step refresh interval for late-stage velocity reutilization. After the full-compute transition at step 19, fresh conditional evaluations occur at steps 20 and 23. At the remaining steps, the sampler reutilizes the latest fresh velocity field without a new DiT evaluation. Every Euler update remains unchanged. This follows the same principle as CFG-on sampling: reutilize velocity predictions while limiting their staleness.

C.1.2 Threshold Expansion: Selection and Fidelity

Selection and speed–fidelity trade-off. Figure 9 varies $\alpha \in \{1, 3, 5\}$ in *TTD* while keeping *CCR* and all other settings fixed. Larger thresholds select more tokens for velocity reutilization, including tokens near the emerging surface. This agrees with the overlapping activity distributions in Figure 8: *TTD* follows token activity, not a fixed non-surface mask. Panel (b) reports Stage-I speedup and occupancy agreement with same-seed full computation on the calibration set. This agreement differs from ground-truth quality on Toys4K-210 in Table 2. The small speedup gain beyond $\alpha = 3$ motivates our default.

Offline trajectory analysis. We next examine whether the surface tokens admitted by threshold expansion also exhibit lower future velocity-reutilization deviation.



(a) Velocity reutilization at $\alpha = 1, 3, 5$.

(b) Speed–fidelity trade-off.

Figure 9: ***TTD* threshold expansion.** (a) Token selection at steps $8, 13, 18$ for $\alpha = 1, 3, 5$. Cyan circles mark tokens selected for velocity reutilization; color denotes guided velocity magnitude, not activity. (b) Stage-I speedup and occupancy IoU against same-seed full compute on the calibration set. Only α varies; *CCR* and all other settings remain fixed.

Protocol and token groups. We analyze the saved full-compute TRELLIS trajectories for all 42 calibration images, using seed 42, CFG strength 5, 25 Euler steps, and timestep rescaling 3. No new inference is required. At each anchor step t, candidates use the stored activity and fixed boundary from Section 3.2. Neither future velocities nor surface labels enter this selection. Surface and non-surface labels follow the positive-occupancy rule defined above, using the final decoded structure. These are labels for spatial token blocks, not pointwise surface labels.

We test $\alpha \in \{1, 2.2, 3, 4, 5\}$ and future horizons $H \in \{1, 2, 3, 4\}$. The primary setting, $\alpha = 3$ and $H = 3$, is fixed before computing the target statistics. Anchors span steps 6–14, and all future steps end by step 18 to avoid the CFG transition. Coverage statistics separately use steps 6–18. We distinguish all selected surface tokens from the surface tokens added beyond $\alpha = 1$, namely $\mathcal{S} \cap [\mathcal{C}_t(\alpha) \setminus \mathcal{C}_t(1)]$. These are activity-based *TTD* candidates on a shared full-compute trajectory, not the masks realized during accelerated sampling with *CCR* and global refreshes.

Future velocity-reutilization deviation. For each token, we compare its anchor velocity with future guided velocities along the same full-compute trajectory. Hypothetical velocity reutilization starts at $t + 1$. With $\Delta_r = |u_{r+1} - u_r|$ and $C = 8$, the integrated absolute deviation is:

$$D^{\text{abs}}_{t,H}(i) = \sum_{j=1}^{H} \Delta_{t+j} \frac{\|\mathbf{v}_{t+j}(i) - \mathbf{v}_t(i)\|_2}{\sqrt{C}}. \tag{16}$$

Summing nonnegative deviations avoids cancellation across steps. We also measure a duration-weighted, symmetric relative vector deviation,

$$D^{\text{rel}}_{t,H}(i) = \frac{\displaystyle\sum_{j=1}^{H} \Delta_{t+j} \frac{2\|\mathbf{v}_{t+j}(i) - \mathbf{v}_t(i)\|_2}{\max(\|\mathbf{v}_{t+j}(i)\|_2 + \|\mathbf{v}_t(i)\|_2, 10^{-8})}}{\displaystyle\sum_{j=1}^{H} \Delta_{t+j}}. \tag{17}$$

The first metric measures absolute update deviation; the second reduces its dependence on velocity scale. Neither measures closed-loop generation error.

Controls and aggregation. The uniform reference is the exact expected mean of a same-count random subset from the surface pool. For the added-surface comparison, both selection and the reference pool exclude $\alpha = 1$ candidates. A second reference matches the selected counts within ten rank bins of initial velocity magnitude. Matching is approximate within each bin and does not enforce identical initial velocity magnitudes.

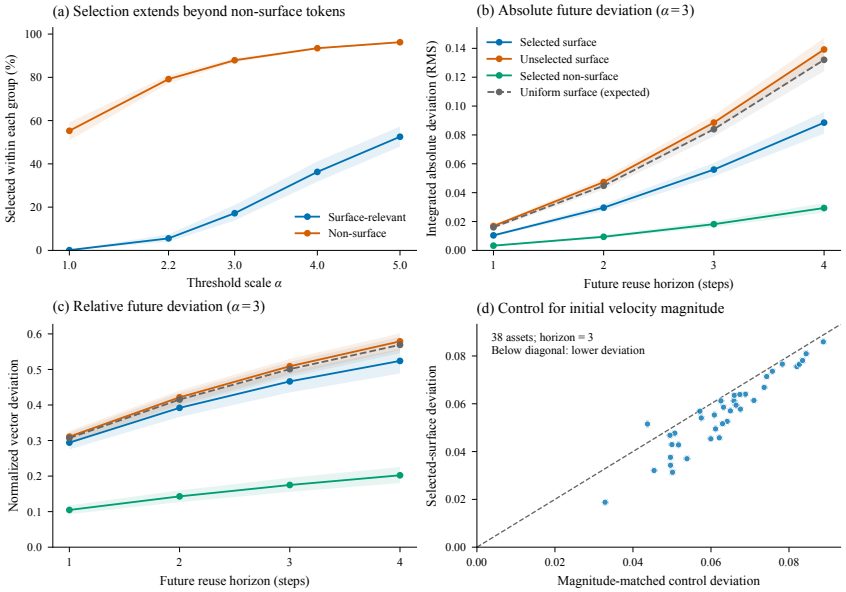


Figure 10: **Threshold expansion selects surface tokens with lower average deviation from velocity reutilization.** Analysis uses full-compute trajectories on the calibration set at CFG 5 and seed 42. (a) Within-group selection rates over steps 6–18, using all 42 assets. (b,c) Absolute and relative deviations at $\alpha = 3$ over future horizons of one to four steps. Curves average asset means; bands are 95% asset-bootstrap intervals. (d) Per-asset absolute deviation at $H = 3$ after initial-magnitude matching; points below the diagonal favor selection. Panels (b–d) use 38 informative assets. These diagnostics use full-compute trajectories and do not measure errors during accelerated generation.

Table 6: **Primary surface-token comparisons at $\alpha = 3$, $H = 3$.** Changes are relative to the stated reference, with paired 95% asset-bootstrap intervals in brackets ($n = 38$). Negative values indicate lower deviation. *Added* excludes candidates already selected at $\alpha = 1$.

Surface group	Reference	Absolute change (%)	Relative change (%)
All selected	Uniform	-33.31 $[-37.27, -29.63]$	-6.89 $[-11.95, -2.13]$
All selected	Magnitude matched	-11.35 $[-14.56, -8.45]$	-11.07 $[-14.49, -7.96]$
Added	Uniform	-33.26 $[-37.25, -29.61]$	-6.81 $[-11.85, -2.08]$
Added	Magnitude matched	-11.32 $[-14.49, -8.43]$	-11.03 $[-14.40, -7.96]$

At each anchor, the selected and unselected surface groups must each contain at least 16 tokens. An asset needs at least three valid common anchors. We average tokens within each anchor, anchors within each asset, and then assets equally. The primary error comparison retains 38 assets. Space station, robot dinosaur, biplane, and helicopter lack sufficient selected surface tokens; all four remain in the 42-asset coverage analysis. Confidence intervals use 2,000 paired bootstrap resamples of assets with seed 42, not resamples of individual tokens or steps. Percentage changes are $100(\overline{D}_{\text{selected}}/\overline{D}_{\text{control}} - 1)$.

Results and scope. With the surface labels defined in Section 3.2, $\alpha = 1, 3, 5$ select 0.13%, 17.17%, and 52.51% of surface tokens, respectively. These percentages use surface tokens as the denominator, not all candidates. Table 6 shows that the surface tokens added by expansion have 33.26% lower absolute deviation than uniform selection, and 11.32% lower deviation after magnitude matching. Both absolute comparisons favor the added set in 37 of 38 assets. Thus, spatial calibration can extend to activity-based selection without restricting velocity reutilization to non-surface tokens.

The result is an average association, not a guarantee for each token or asset. The normalized comparison against the uniform reference favors only 20 of 38 assets. For all selected surface tokens, the

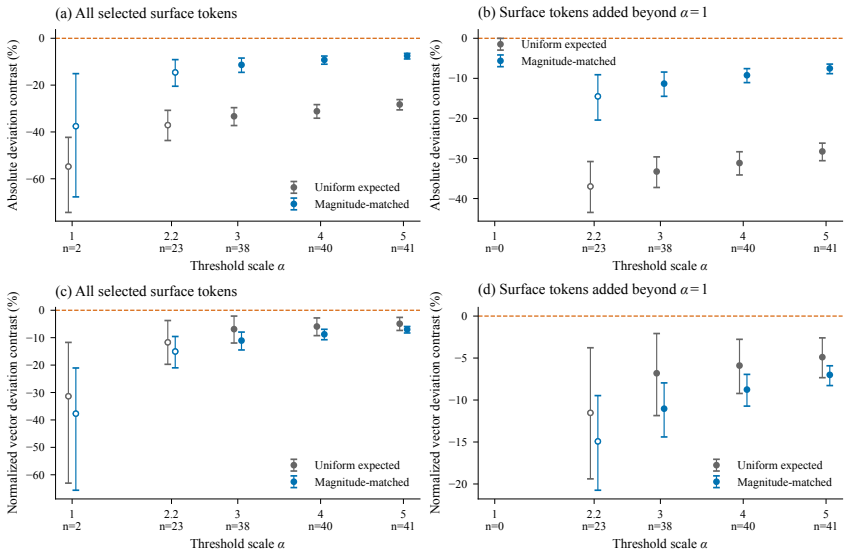


Figure 11: **Sensitivity of the offline velocity-reutilization diagnostic to threshold expansion.** At $H = 3$, panels compare all selected surface tokens and those added beyond $\alpha = 1$, using absolute and relative deviation. Points show percentage changes from uniform or magnitude-matched references; bars are paired 95% asset-bootstrap intervals. Effective asset counts are printed below each α; hollow points have fewer than 30 assets. The cohorts differ across α, so these are descriptive sensitivity checks, not paired cross-α effects. The $\alpha = 1$ surface estimate uses only two assets.

one-step normalized contrast has a 95% interval of $[-9.60, +0.90]\%$, and the three-step directional-variation contrast has an interval of $[-13.24, +2.24]\%$; neither establishes a reduction. We therefore claim lower average velocity-reutilization deviation, not uniformly stable directions. These examples also informed calibration, so this is not held-out generalization evidence. Recomputing the retrospective labels with the rule in Section 3.2 leaves the fitted runtime boundary unchanged and preserves the directions of the tested contrasts. Source trajectories and runtime settings are unchanged. Speed, final quality, and contextual effects are assessed separately by the online ablations and context interventions.

C.2 Cached Context Restoration (CCR)

Preserving the full context. *CCR* follows Eq. 3. Fresh rows compute new queries, keys, and values. Reused rows supply cached keys and values at their original token positions. Fresh queries attend to the full spatial context, which combines fresh and cached K/V rather than recomputing every entry. Cross-attention, the MLP, and the remaining block operations run only on fresh rows. Cache writes replace the corresponding rows; they do not append temporal copies. Conditional and unconditional branches use separate caches.

Fresh output velocities are merged with cached velocities before the native CFG combination. The Euler solver then updates every latent token. *TTD* with *CCR* therefore updates all latent states while retaining full self-attention context. A global refresh recomputes all token rows, updates both caches, and reassesses activity. Restoring cached context and refreshing it are distinct operations.

Cache refresh scheduling. All latent tokens continue to evolve through Euler updates, so cached K/V states and velocity predictions can become outdated. *CCR* uses elapsed sampling steps as a low-cost proxy for cache staleness. Let r be the latest global refresh step, $g_t = t - r$, and H the accumulated score. Before a decision, we compute

$$H_t^{\text{prop}} = H + \min\left(\frac{g_t}{K}, 1\right). \tag{18}$$

The main setting is $K_{\min} = 2$, $K = 5$, and $\delta = 1.21$, where K is the maximum refresh interval defined in Section 3.3. The schedule starts at step 6 and uses boundaries $\mathcal{B} = \{0, 6, 19\}$. Let d_t be

Table 7: Sensitivity to K on Toys4K-210, with 210 assets per configuration. Refresh steps are zero-based and include cache initialization at step 6. The shared transition step 19 and CFG-off full-compute steps 20 and 23 are omitted.

K	CFG-on refresh steps	Speedup ↑	FLOPs (T) ↓	vIoU ↑	CD ↓	F-score ↑	PSNR ↑	LPIPS ↓
Baseline	–	1.000	244.24	61.03	0.02785	95.93	20.662	0.0959
3	6, 9, 12, 15, 17	1.296	156.73	61.14	0.02788	95.97	20.664	0.0963
4	6, 9, 12, 15	1.331	147.06	60.88	0.02785	95.97	20.609	0.0965
5 (default)	6, 10, 14	1.342	138.03	60.65	0.02777	95.99	20.636	0.0966
10	6, 11, 16	1.368	138.40	61.19	0.02785	95.99	20.617	0.0963

the number of steps to the next boundary, or infinity if none remains. The implementation refines the main-text rule near mandatory boundaries, checking the following conditions in order:

1. Refresh if the cache is uninitialized or t is a boundary.
2. Refresh if $d_t = K_{\min}$ and $g_t = K - 1$ to align with a nearby boundary.
3. Refresh if $g_t \geq K$.
4. Otherwise, refresh if $g_t \geq K_{\min}$ and $H_t^{\text{prop}} > \delta$, except when $0 < d_t < K_{\min}$.

A refresh recomputes all token rows, updates both K/V and velocity caches, reevaluates token activity, and sets $r \leftarrow t$ and $H \leftarrow 0$. Otherwise, $H \leftarrow H_t^{\text{prop}}$. Each reused entry must also satisfy $t - \rho_t(i) < K$, where $\rho_t(i)$ is the latest step at which token i received a new DiT prediction. These are global refresh events, not independent decisions to refresh individual tokens. Tokens outside $\mathcal{R}_t$ still receive new DiT predictions at each step.

Parameter roles and realized schedule. $K_{\min}$ is the minimum interval before the score can trigger a refresh. K normalizes the elapsed steps and sets the maximum interval. δ controls the accumulated score threshold. Boundary rules avoid an unnecessary refresh immediately before a mandatory one. Given these parameters and boundaries, the schedule is deterministic. It does not adapt to measured prediction errors or image content.

The default cache initialization and global refresh steps are 6, 10, 14, and 19. After a refresh, the first three increments sum to $0.2 + 0.4 + 0.6 = 1.2$, below $\delta = 1.21$. The fourth crosses the threshold. Near the CFG transition, the step-18 crossing is deferred to step 19. Thus, $K = 5$ does not imply a fixed five-step period, and $K_{\min} = 2$ does not imply a two-step refresh. The schedule limits cache staleness without claiming an optimal refresh time.

C.2.1 Sensitivity to K

We evaluate $K \in \{3, 4, 5, 10\}$ on Toys4K-210 while fixing $\alpha = 3$, immediate admission, $K_{\min} = 2$, and $\delta = 1.21$. Activity calibration, warmup, and CFG-off sampling remain unchanged. This ablation varies K in the refresh controller: it affects both normalization of elapsed steps and the maximum refresh interval, rather than specifying a fixed refresh period. Each global refresh jointly updates velocities and per-block K/V caches, so this is not an test that isolates K/V staleness.

Table 7 shows that quality does not vary monotonically with K. In particular, $K = 5$ and 10 produce the same number of CFG-on full-compute steps but place them differently. Their differences therefore cannot be explained solely by refresh count. The larger setting yields higher vIoU, whereas the default yields lower CD and higher PSNR. We retain $K = 5$ as the default and interpret these results as controller sensitivity rather than evidence for a universally optimal interval.

Timing and reference reuse. The baseline and $K = 5$ results are reused from existing runs. For the other settings, each asset uses one untimed warmup and three synchronized repetitions with rotated method order. We report the mean per-asset speedup relative to the stored baseline latency, using each new configuration's median latency. These are not newly interleaved timings across all configurations. FLOPs cover the Stage-I Transformer core.

C.3 Controlled K/V Context Interventions

Question and controlled variables. This experiment tests the motivation for *CCR*: whether tokens selected by *TTD* for velocity reutilization still affect velocity predictions on active tokens through their K/V context. It separates this question from the end-to-end refresh ablation in Section 4.3, where a full refresh jointly updates velocities and K/V caches. Here, the reused-token velocities are identical across all three treatments. Only the source of their K/V context changes.

Assets and sampling states. We select 16 assets from the calibration set by sorting the SHA-256 hashes of the string `fixed-context-v1/` followed by each asset identifier, and taking the first 16. The subset contains rock monster, block robot, excavator, cart, workbench, helicopter, T, elephant, magic book, space station, colorful cottage, lantern, paper machine, television, castle, and mech. Selection does not use intervention outcomes.

We use the main TRELLIS Faster3D configuration with $\alpha = 3$, immediate admission, and the refresh controller with $(K_{\min}, K, \delta) = (2, 5, 1.21)$. Sampling uses seed 42, 25 Euler steps, timestep rescaling 3, and CFG strength 5. Each intermediate state is taken from this same accelerated reference trajectory. We probe zero-based steps 8, 12, and 16 during CFG-on sampling. Step 12 was evaluated first; steps 8 and 16 were subsequently added as a robustness check, with the asset list, seed, and new steps fixed before the extension. Thus, the study contains 48 paired events on 16 assets, not 48 independent assets or a new held-out dataset.

Three context treatments. At step t, let $\mathcal{R}_t$ denote the indices of tokens selected for velocity reutilization and $\mathcal{F}_t$ the remaining token indices, as in Section 3.3. We call tokens in $\mathcal{F}_t$ active tokens because they receive new DiT predictions. All treatments share the input latent, timestep, condition, model weights, token sets, and cached guided velocity on $\mathcal{R}_t$:

- **Removed context:** active queries attend only to the newly computed K/V rows of $\mathcal{F}_t$; rows from $\mathcal{R}_t$ are absent.
- **Cached context:** active queries attend to new K/V rows of $\mathcal{F}_t$ and the historical K/V rows of $\mathcal{R}_t$, restored at their original token positions.
- **Fresh oracle context:** a full evaluation at the same event input supplies current K/V rows for $\mathcal{R}_t$ at each self-attention block. These replace the historical rows. The treatment computes its own active rows and still uses the same cached output velocities on $\mathcal{R}_t$.

The newly predicted oracle velocities for reused tokens are discarded. Conditional and unconditional branches keep separate K/V caches, and cross-attention is unchanged. The intervention is applied to all 24 self-attention blocks. Active hidden states and queries at later blocks may differ as a consequence of the changed context; they are not artificially held fixed. This propagation is part of the contextual effect being measured.

Immediate and terminal measurements. Let $\mathbf{v}_t^{\text{full}}$ be the guided velocity from a full evaluation at the same event input. For treatment h, we measure the RMS deviation on active tokens:

$$D_{t,h} = \left(\frac{1}{|\mathcal{F}_t| C} \sum_{i \in \mathcal{F}_t} \|\mathbf{v}_{t,h,i} - \mathbf{v}_{t,i}^{\text{full}}\|_2^2 \right)^{1/2}, \qquad C = 8. \tag{19}$$

This measures the change to tokens that are still being computed, rather than the directly reused velocities. The low oracle deviation is partly expected because its context comes from the same full computation. The terminal comparison tests whether this local intervention also affects the generated structure.

For each event, we take one Euler step with the treatment velocity and then run full computation at every remaining step, with the same timestep and CFG settings. No later *TTD* selection or *CCR* operations are applied. Each continuation contains exactly one intervention; the three probe times start separate trajectories. A fourth continuation uses the full-current velocity on all tokens at the event and then the same full-compute continuation. It is the common fidelity reference, not another treatment that holds reused velocities fixed. It starts from the same accelerated intermediate state, not from the original input noise.

Table 8: **Context interventions at fixed reused velocities.** Mean averages steps 8, 12, and 16 within each asset, then averages 16 assets; individual-step blocks average the same assets. Each terminal trajectory has one intervention followed by full computation. Active velocity RMS and occupancy IoU use the full-compute references defined in this section, not ground truth. Fresh oracle is diagnostic.

Step	Context	Active velocity RMS ↓	Occupancy IoU (%) ↑
Mean	Removed	2.9197	94.49
	Cached	0.4251	98.46
	Fresh oracle	0.0044	99.37
8	Removed	2.9917	91.37
8	Cached	0.3871	97.98
8	Fresh oracle	0.0020	99.37
12	Removed	3.1640	94.97
12	Cached	0.3831	98.71
12	Fresh oracle	0.0037	99.39
16	Removed	2.6034	97.12
16	Cached	0.5051	98.67
16	Fresh oracle	0.0075	99.36

We decode each terminal sparse latent into a 64^3 occupancy grid using a decoder-logit threshold of zero. For treatment occupancy O_h and reference occupancy O_{ref}, we report $|O_h \cap O_{\mathrm{ref}}|/|O_h \cup O_{\mathrm{ref}}|$. This is agreement with the controlled full-compute reference, not ground-truth vIoU. No texture or rendered-image metric is used in this diagnostic.

Aggregation and paired effects. We first average the three times within each asset, then average the 16 asset means. For confidence intervals, we draw 2,000 paired bootstrap samples of assets with seed 42, retaining all times and treatments of each sampled asset. Retaining cached context improves mean terminal agreement by 3.9691 percentage points, with a 95% interval of $[3.2480, 4.7624]$. Fresh context adds 0.9175 points, with interval $[0.6841, 1.1912]$. Both contrasts are positive for all 16 asset means.

At the event level, cached context improves agreement over removal in all 48 events. Fresh context improves over cached context in 47 of 48 events; lantern at step 12 decreases by 0.0894 percentage points. Table 8 reports the overall and per-time means. Because earlier interventions have longer remaining trajectories, their larger terminal effects alone do not establish an optimal refresh time. The intervals describe variation across these assets at one seed, not variation across seeds or backbones.

Implementation checks and interpretation. We verify that diagnostic probes leave the accelerated reference trajectory unchanged at all 25 steps. Event inputs, masks, and reused-token velocities must match exactly across treatments. The cached-context treatment must reproduce the production computation, and terminal metrics are recomputed from saved occupancy arrays. Input images, source code, parent states, and reused step-12 results are fingerprinted. All 16 assets pass these checks, with no empty terminal occupancies.

These controls attribute the measured differences to the K/V intervention within the tested setting. They support retaining K/V context through *CCR* when *TTD* reutilizes token velocities. The fresh-context comparison supports refreshing K/V, but does not evaluate the full refresh policy, which also updates cached velocities. The fresh oracle requires full computation and serves only as a diagnostic reference, not an acceleration method. End-to-end quality and acceleration remain measured by the main comparison and ablation tables.

C.4 Cross-Method Cached Context Restoration

Protocol and controls. We apply the original-position K/V restoration used by *CCR* to Fast3Dcache, without replacing its token selector or adding the Faster3D refresh controller. All five settings in Table 9 use the same 210 Toys4K inputs, TRELLIS weights, seed 42, 25-step Stage-I

Table 9: Cross-method cached K/V restoration on Toys4K-210. Reference speedups follow Tables 1 and 3. ‡ marks a separate timing batch using a historical full-compute reference, so speedups are not a contemporaneously matched comparison. FLOPs count the Stage-I Transformer core. vIoU and F-score are percentages.

Method	Speedup ↑	FLOPs (T) ↓	PSNR ↑	LPIPS ↓	vIoU ↑	CD ↓	F-score ↑
Full compute	1.000	244.24	20.662	0.0959	61.03	0.02785	95.93
Fast3Dcache	1.338	132.43	20.270	0.1007	57.38	0.02838	95.93
+ cached K/V	1.259‡	134.01	20.575	0.0964	61.18	0.02791	95.96
TTD-only	1.466	132.69	20.335	0.0993	58.17	0.03014	95.07
Faster3D	1.342	138.03	20.636	0.0966	60.65	0.02777	95.99

sampler, CFG scale 5, and full-compute Stage II. No hyperparameters are searched in this experiment. Existing full-compute, Fast3Dcache, and Faster3D quality results are reused after checking sample identities, input hashes, and recorded configurations. Only Fast3Dcache with cached K/V restoration requires new assets. The Faster3D pair uses $\alpha = 3$, immediate admission, and the refresh settings $(K_{\min}, K, \delta) = (2, 5, 1.21)$. Its control is ***TTD-only***, as in Table 3: it retains cached velocities, Euler updates for all tokens, and the normal refresh schedule, but omits cached K/V restoration.

Cached K/V restoration. The Fast3Dcache pair shares the original PCSC/SSC implementation, velocity cache, Euler integration, warmup, CFG transition, and native correction rules. Full evaluations starting at the existing anchor step 5 capture K/V for all 24 self-attention blocks, with separate conditional and unconditional caches. During mixed evaluations, only the selected fresh rows produce new Q/K/V. Their K/V overwrite the corresponding original cache rows, while omitted rows retain their most recent K/V, giving a full 4096-token attention context. Image cross-attention is unchanged. No additional full evaluations or global refresh policy are introduced: SSC-triggered recomputation and the native late corrections remain in place. Thus, this is Fast3Dcache plus K/V restoration, not plus the entire Faster3D controller.

Metrics and timing. The geometry and appearance protocols follow Appendix B; all 210 assets have valid scores for all five quality metrics, with no dropped samples. For the new Fast3Dcache+K/V run, Stage I timing uses one warmup and three CUDA-synchronized repetitions per asset, taking the median. Its reported speedup divides the historical full-compute mean latency by the mean of these per-asset medians. Baseline timing is not rerun or interleaved with this added variant. The other reference speedups are taken from the main text. Because the added variant uses a separate timing batch and aggregation rule, this table supports the quality comparison but does not establish superiority at matched runtime. Appendix C.5 provides a separate comparison with contemporaneous timing. Wall time includes online PCSC decoding and restoration overhead, whereas core FLOPs exclude decoder work and memory traffic.

Results and interpretation. For Fast3Dcache, cached K/V restoration increases mean vIoU by 3.80 percentage points and PSNR by 0.305 dB, while reducing LPIPS by 0.00430. The paired medians are $+0.246$ vIoU points and -0.00257 LPIPS, respectively; $121/210$ assets improve in vIoU and $157/210$ in LPIPS. The benefit is therefore not uniform across assets. This is a closed-loop comparison: preserving context can change later velocities, selected masks, and realized refresh events even when the selection rules stay fixed. It supports context preservation under two selection policies, but does not isolate a fixed-mask causal effect or establish selector superiority at matched compute. The single-state interventions in Appendix C.3 address the former question separately.

C.5 Token Selection at Comparable Runtime

Comparison protocol. We compare *TTD* at $\alpha = 3$ with the PCSC budget and velocity/acceleration ranking of Fast3Dcache under shared *CCR* and refresh rules. Both use TRELLIS with seed 42, 25 steps per stage, CFG strength 5, timestep rescaling 3, and full Stage II computation. Warmup, cache updates, refreshes, and the CFG-off schedule are identical. Only freshly evaluated

tokens can become newly eligible for velocity reutilization. The control uses acceleration weight 0.7 and budget multiplier β, with shared rules replacing native cache expiry and correction. This is a selection control, not original Fast3Dcache or the variant in Appendix C.4.

Runtime matching. We test ten β values from 0.15 to 1.2 on 12 calibration images and confirm the selected configurations on all 42, disjoint from Toys4K-210. Matching uses only mean Stage I latency within a prescribed 3% tolerance, without quality metrics. The reported pair, $\alpha = 3$ and $\beta = 1$, is fixed before testing. Toys4K-210 timings use rotated configuration order, one warmup and three synchronized repetitions per asset. We average per-asset median latencies, including selection, online PCSC decoding, K/V restoration, and refreshes, but excluding Stage II, export, and evaluation.

All 210 assets have valid scores for the five metrics in Appendix B. We reuse *TTD* outputs after verifying input identities, configurations, seeds, and sampling settings, and generate new control outputs. The experimental implementation reproduces all 25 production steps on three calibration assets. Paired intervals use 2,000 bootstrap resamples of assets.

Table 10: Token selection with shared *CCR* on Toys4K-210 ($\alpha = 3$). Time is mean Stage I latency (s). vIoU and F-score are percentages. The runtime gap is 1.36% relative to *TTD* using unrounded latencies, within the 3% tolerance.

Selection	Time ↓	PSNR ↑	LPIPS ↓	vIoU ↑	CD ↓	F-score ↑
TTD ($\alpha = 3$)	0.799	20.636	0.09661	60.65	0.02777	95.99
Fast3Dcache selection ($\beta = 1$)	0.810	20.553	0.09719	60.69	0.02804	95.94

Results and scope. At $\alpha = 3$, *TTD* improves CD and PSNR at comparable runtime (Table 10). Mean differences from the control are -0.000266 for CD (95% interval $[-0.000449, -0.000067]$) and $+0.0826$ dB for PSNR ($[0.0124, 0.1531]$). Intervals for vIoU, F-score, and LPIPS include zero. This supports *TTD* selection under shared *CCR* at the default setting, not gains on every metric or across all α. The comparison tests the score and selection rules together, not the score alone, on TRELLIS at one seed.

D Additional Qualitative Results

Figures 12–16 compare **Faster3D** with the diverse image-to-3D backbones.

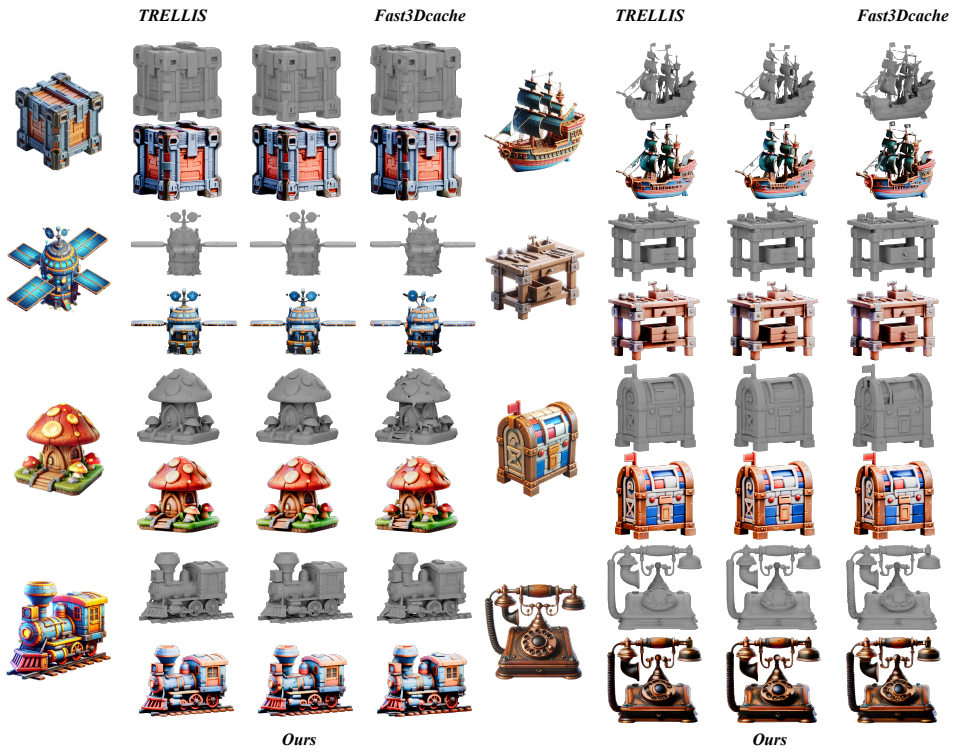


Figure 12: **Additional qualitative comparisons on TRELLIS.** Untextured and textured renderings compare full-compute TRELLIS, **Faster3D**, and Fast3Dcache.

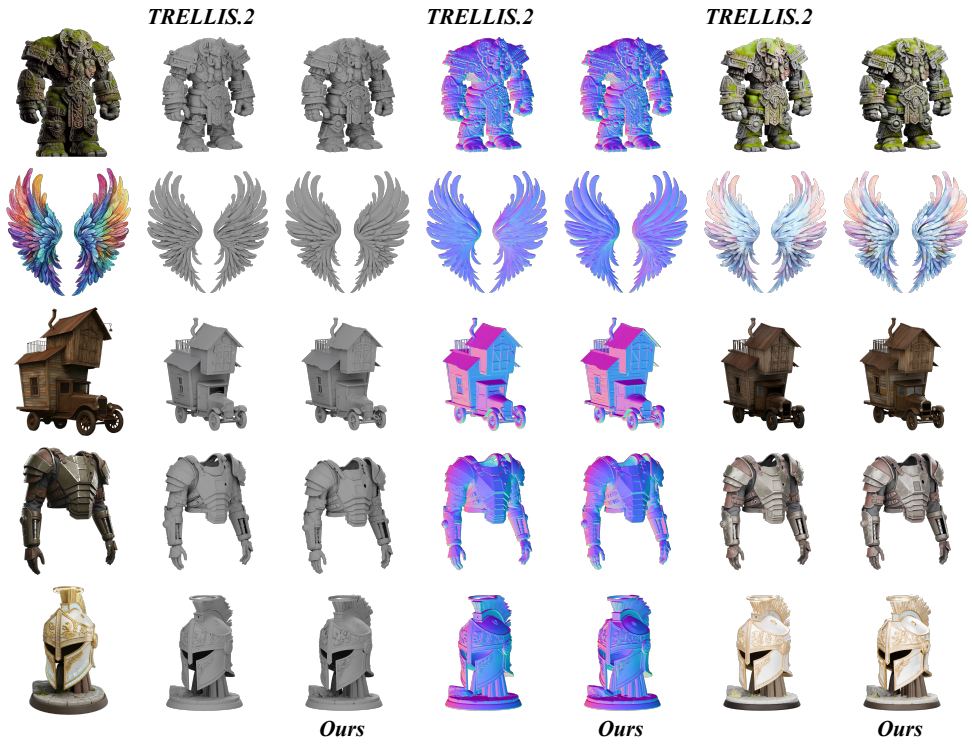


Figure 13: **Qualitative transfer to TRELLIS.2.** Untextured, surface-normal, and textured renderings compare full-compute TRELLIS.2 and **Faster3D**.

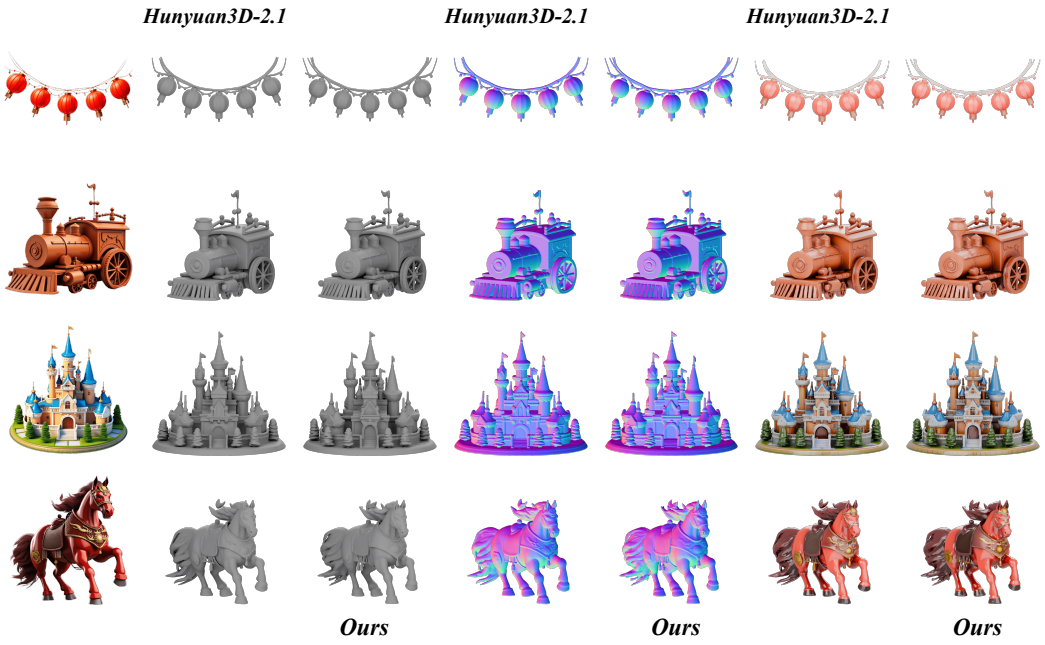


Figure 14: **Qualitative transfer to Hunyuan3D-2.1.** Each row compares full-compute Hunyuan3D-2.1 and **Faster3D** using paired untextured, surface-normal, and textured renderings.

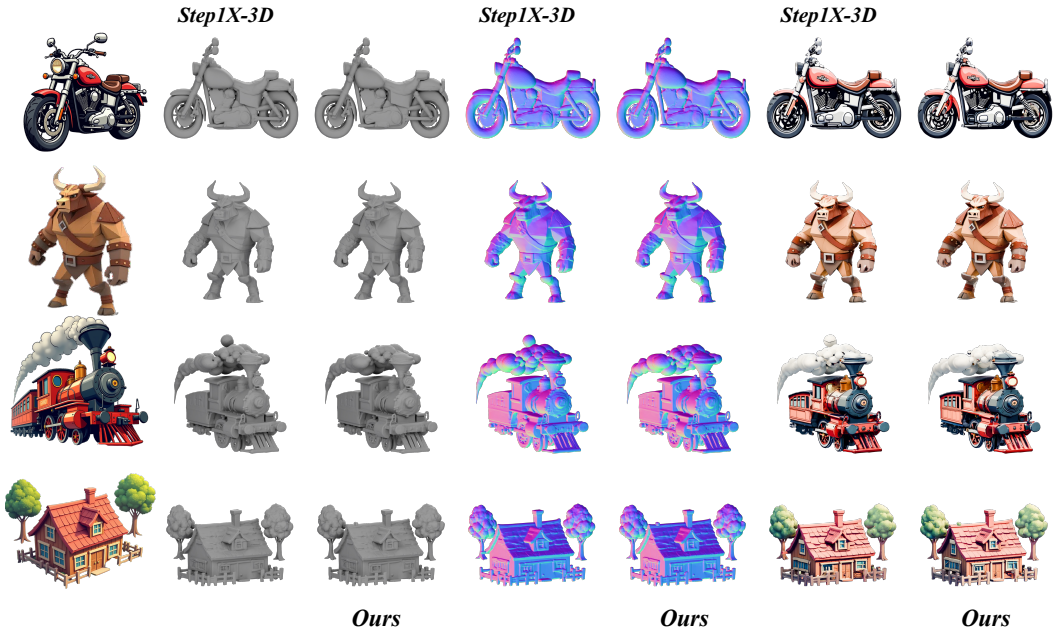


Figure 15: **Qualitative transfer to Step1X-3D.** Each row compares full-compute Step1X-3D and **Faster3D** using paired untextured, surface-normal, and textured renderings.

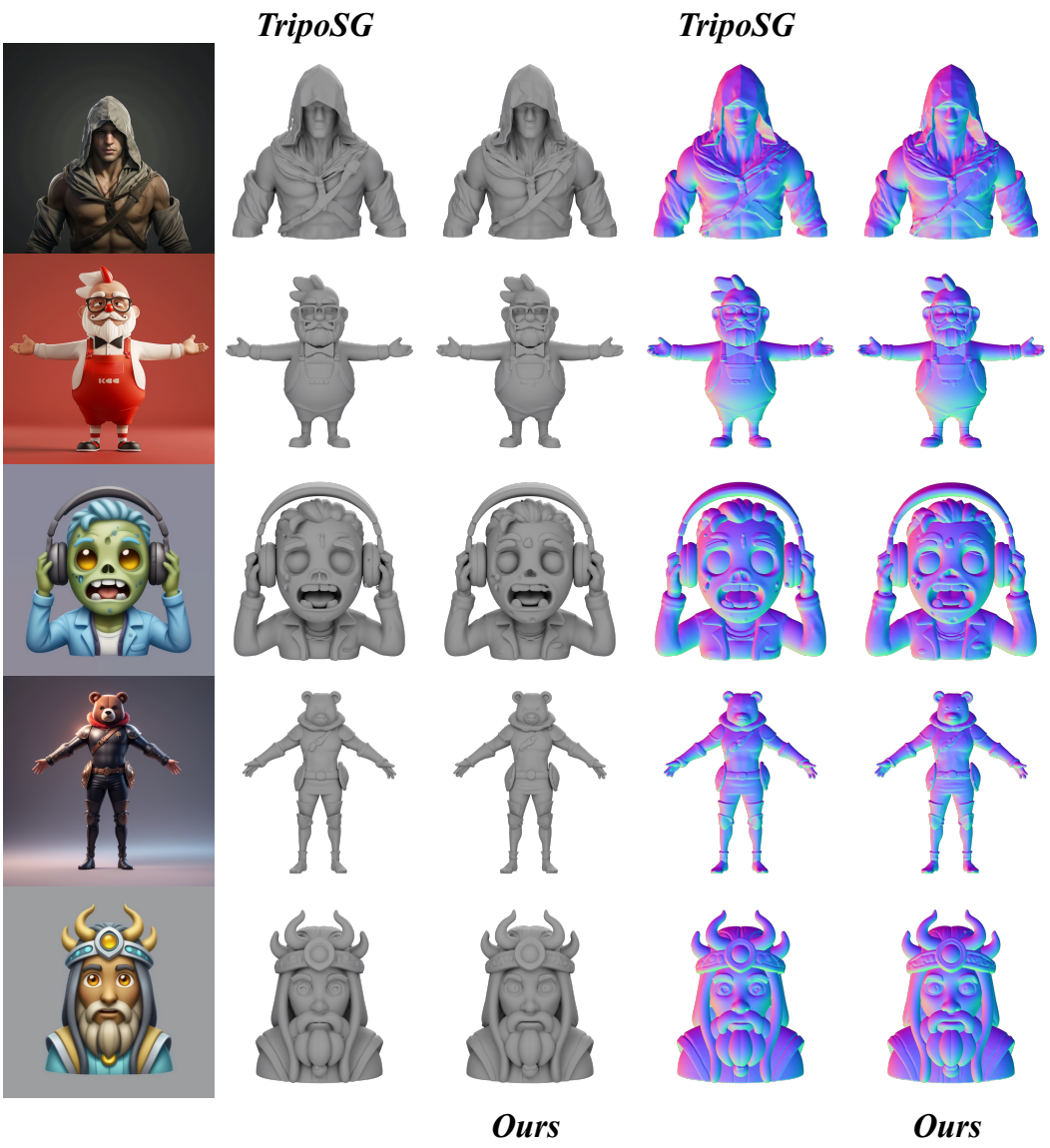


Figure 16: **Qualitative transfer to TripoSG.** Each row shows an input followed by paired untextured and surface-normal renderings of full-compute TripoSG and **Faster3D**. TripoSG generates geometry only.