R3D: RESIDUAL RELIABILITY ROUTING FOR TRAINING-FREE IMAGE-TO-3D ACCELERATION

Anonymous authors
Paper under double-blind review



Figure 1: **R3D across four 3D backbones.** Qualitative comparisons of baseline and **R3D** outputs given the same input images. Red labels indicate speedups in texture generation stage.

ABSTRACT

Recent diffusion- and flow-based image-to-3D generators synthesize high-quality 3D assets, but remain computationally expensive due to repeated Transformer computation throughout sampling. Existing training-free acceleration methods reduce this cost by exploiting temporal, stage-wise, or token-wise redundancy, yet they do not characterize whether the residual updates from Transformer blocks can be reliably predicted for faster inference. In this paper, we first characterize the reliability of predicting these residual updates within the texture stage. We find that prediction reliability varies substantially across sampling time and network depth, while the relative ordering of reliable and unreliable locations remains consistent across inputs. Moreover, fine-grained residual units within a Transformer block can remain predictable even when whole-block residual prediction is unreliable. Motivated by these findings, we propose **Residual Reliability Routing** (**R3D**), a training-free framework that uses the reliability of residual prediction to construct a reusable routing policy. **R3D** first constructs reusable reliability profiles from a small calibration set through *Residual Reliability Profiling (RRP)*. Then, we propose *Granularity-Aware Routing (GAR)*, which selects the residual unit within a Transformer block that reduces the most computation. Finally, *Exact-Anchor Refresh (EAR)* recomputes selected residual units in the Transformer block to refresh prediction histories. The resulting routing policy is fixed before inference and generalized to unseen inputs without modifying the pretrained generator. Experimental results show that **R3D** consistently reduces texture stage inference cost while largely preserving 3D generation quality.

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

Diffusion models (Ho et al., 2020) and flow-matching models (Lipman et al., 2023) have substantially advanced image-to-3D generation. Modern systems such as TRELLIS (Xiang et al., 2025b) and Hunyuan3D-2.1 (Hunyuan3D et al., 2025) commonly adopt multi-stage pipelines that first establish spatial structure and then synthesize finer geometric and appearance details. Despite their strong generation quality, these systems remain computationally expensive because of repeated computation through large Diffusion Transformer (DiT) backbones (Peebles & Xie, 2023). Reducing this repeated computation therefore becomes an important problem for efficient image-to-3D synthesis.

Existing approaches alleviate this cost through either training-based or training-free approximation. Training-based methods (Liu et al., 2025d; Hu et al., 2025) reduce sampling cost through distillation or few-step adaptation, but require additional optimization. Training-free approaches instead exploit redundancy already present in pretrained generators. For example, Fast-SAM3D (Feng et al., 2026) exploits heterogeneous dynamics across generation stages and spatial regions. Fast3Dcache (Yang et al., 2026) adapts caching during the shape stage according to temporal geometric stabilization and can leave structural differences in the selected examples in Fig. 2. These methods commonly establish that computation in image-to-3D generation is substantially redundant and that such redundancy is non-uniform. However, redundancy alone does not indicate whether a skipped residual computation can be reliably predicted. This motivates a complementary question: *does the reliability of residual prediction itself exhibit reusable structure across the sampling process?*

We study this problem in the texture stage of image-to-3D pipelines, a computationally substantial target for acceleration. Our empirical analysis on texture stage reveals three characteristic properties of residual prediction reliability. First, *reliability is location-dependent*: residual prediction errors vary considerably across sampling time and network depth, revealing a non-uniform reliability landscape over computation locations; second, *reliability structure is general*: although absolute prediction errors vary across objects, the relative ordering of reliable and unreliable locations remains consistent across inputs, enabling reliability profiles estimated from a small calibration set to be reused for unseen inputs; third, *predictability is recoverable at finer granularity*: even when whole-block prediction is unreliable, finer residual units from the same block can remain reliably predictable. Together, these findings suggest that residual prediction should be allocated according to its measured reliability, rather than inferred from computational redundancy alone.

All these observations motivate **R3D**, a training-free ***R**esidual **R**eliability **R**outing* framework for image-to-3D acceleration. The **R3D** framework is composed of three components. First, *Residual Reliability Profiling (RRP)* estimates location- and granularity-specific reliability from a small calibration set, and exploits its cross-input consistency to construct a reusable reliability profile. Second, *Granularity-Aware Routing (GAR)* then selects the coarsest admissible residual unit that satisfies the reliability criterion, while retaining exact computation when necessary. Last, *Exact-Anchor Refresh (EAR)* maintains compatible exact anchors and prevents predicted residuals from recursively entering the prediction history. Together, these components compile an offline routing policy that is fixed for each model and inference configuration. At inference time, the same route is reused across inputs, while hidden states and residual histories remain input-specific. Experiments on Toys4K (Stojanov et al., 2021) across TRELLIS, TRELLIS.2 (Xiang et al., 2025a), LumiTex (Bao et al., 2025), and MatLat (Yeo et al., 2026) demonstrate that **R3D** consistently reduces texture stage inference cost while largely preserving generation quality. Our contributions include:

- We show that residual prediction reliability is *location-dependent* across sampling time and network depth, its relative structure is *shared across inputs*, and predictability can be *recovered at finer granularity* when whole-block prediction fails.
- We introduce **R3D**, a training-free *Residual Reliability Routing* framework that converts this structured reliability into reusable compute allocation. It comprises three components:*Residual Reliability Profiling* (RRP), *Granularity-Aware Routing* (GAR), and *Exact-Anchor Refresh* (EAR).
- Extensive experiments on Toys4K demonstrate that **R3D** provides consistent texture stage acceleration across TRELLIS, TRELLIS.2, LumiTex, and MatLat while largely preserving full-compute quality. Ablation studies further validate the structured reliability hypothesis and the individual routing mechanisms.



Figure 2: **Structural sensitivity to shape-stage acceleration.** Selected comparisons between *Fast3Dcache* and original *Native* inference when acceleration is applied only to the shape stage under a conservative setting yielding $1.03\times$ speedup; the subsequent stage is executed with identical full computation for both runs.

2 Related Work

2.1 3D Generative Model

Early 3D reconstruction methods focused on recovering explicit or implicit 3D representations, including point clouds, meshes, and neural fields, from visual observations (Fan et al., 2017; Wang et al., 2018; Mescheder et al., 2019). More recent feed-forward reconstruction models substantially improve inference efficiency and generalization to in-the-wild images (Hong et al., 2024). In parallel, generative approaches introduce learned priors to model the ambiguity of image-conditioned 3D synthesis. Early methods leveraged pretrained 2D diffusion priors for optimization-based 3D generation (Poole et al., 2023), while subsequent work generated geometrically consistent multi-view observations before reconstruction (Long et al., 2024; Liu et al., 2024). With the growth of large-scale 3D data, recent systems increasingly learn generative priors directly over 3D representations or latent spaces. CLAY (Zhang et al., 2024), TRELLIS (Xiang et al., 2025b), and Hunyuan3D (Hunyuan3D et al., 2025) exemplify this trend, combining learned 3D representations with large diffusion- or flow-based generative backbones.

2.2 Training-Free Acceleration of Diffusion and Flow Models

Training-free acceleration operates at both the sampling and network-computation levels. Sampler-based methods such as DDIM (Song et al., 2020) and DPM-Solver (Lu et al., 2022) reduce the number of network evaluations, while network-level methods exploit redundancy across repeated evaluations. Faster Diffusion (Li et al., 2024) reveals component-dependent temporal redundancy, and subsequent caching methods reuse intermediate activations across nearby sampling steps. SmoothCache (Liu et al., 2025c) profiles layer-wise representation errors to construct reusable caching schedules, whereas TeaCache (Liu et al., 2025a) estimates temporal variation online for adaptive recomputation. Beyond direct reuse, TaylorSeer (Liu et al., 2025b) extrapolates intermediate features from their temporal trajectories, providing a prediction-based alternative to direct cache reuse.

Training-free acceleration has also been studied in 3D generation. Hash3D (Yang et al., 2025) exploits temporal and view-related redundancy during 3D optimization. Fast3Dcache (Yang et al., 2026) leverages temporal stabilization of 3D representations for predictive caching, while FastSAM3D (Feng et al., 2026) adapts computation to heterogeneous dynamics across generation stages, modalities, spatial tokens, and object instances. These works establish substantial and non-uniform redundancy in image-to-3D generation. Our work studies a complementary question: whether the *reliability of residual prediction itself* exhibits a reusable structure across diverse dimensions.

3 Method

3.1 Preliminaries

Generation pipeline. Modern image-to-3D generation pipelines (Xiang et al., 2025b;a; Hunyuan3D et al., 2025) commonly decompose generation into multiple stages. We use TRELLIS as the running example below. Its shape stage uses a Flow Transformer to generate a sparse 3D

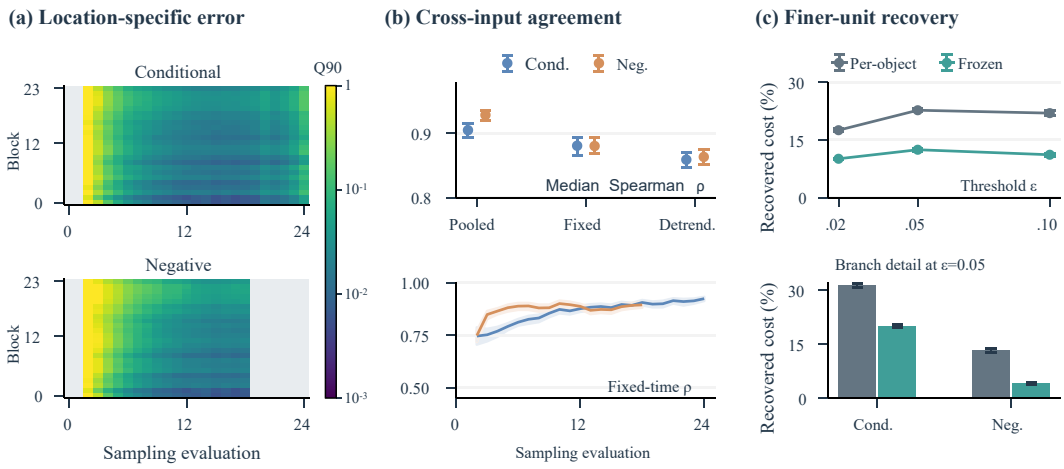

Figure 3: **Three observations supporting reliability routing.** (a) Q90 whole-block relative error across 105 evaluated objects (one per category) depending on locations. (b) Cross-object Spearman agreement: pooled, fixed-time, and detrended summaries (top), and fixed-time agreement by sampling evaluation (bottom). (c) Mean component recovery within whole-block-failure locations under H2: per-object oracle versus choices frozen using 20 disjoint calibration objects; threshold sweep (top) and branch breakdown at $\epsilon = 0.05$ (bottom).

structure that determines the active voxel locations. Conditioned on this structure, the texture stage employs a Sparse Flow Transformer to generate local Structured LATent (SLAT) features at these locations. In this work, our analysis and acceleration are applied to the Sparse Flow Transformer within the texture stage, while the preceding shape stage is kept unchanged.

Residual computation and prediction. At the k-th flow-model computation, the Sparse Flow Transformer operates on hidden features associated with the active sparse voxel locations. Let τ_k denote the corresponding model time, and let $H_{k,\ell}$ and $H_{k,\ell+1}$ denote the input and output hidden features of its ℓ-th Transformer block, respectively. We define the whole-block residual update as $R_{k,\ell} = H_{k,\ell+1} - H_{k,\ell}$, such that $H_{k,\ell+1} = H_{k,\ell} + R_{k,\ell}$. We refer to (k, ℓ) as a *computation location*, jointly specifying the flow-model computation and network depth. Here, $R_{k,\ell}$ represents the hidden-feature update produced by a Sparse Flow Transformer block, rather than the flow velocity between sampling states.

To study whether these residuals can be predicted from their temporal history, we use a naive linear extrapolator. Given two preceding exact residuals $R_{a,\ell}$ and $R_{b,\ell}$ observed at model times τ_a and τ_b, with $a < b < k$, we predict the residual at the current model time τ_k as

$$\widehat{R}_{k,\ell} = R_{b,\ell} + \frac{\tau_k - \tau_b}{\tau_b - \tau_a} \left(R_{b,\ell} - R_{a,\ell} \right). \tag{1}$$

3.2 Empirical Analysis of Residual Reliability

With the definition for residual predictions above, we next examine whether its reliability exhibits regular patterns. We collect residual trajectories from full inference trajectories on image-to-3D pipelines and evaluate the predictor formulated in Eq. (1). We examine prediction reliability along three dimensions: computation location, input object, and residual granularity. For cross-object comparisons, all inputs follow the same anchor-selection rule.

Observation 1: reliability is location-dependent. We first examine whether residual prediction errors are uniform across the sampling locations. For each sampling evaluation and Transformer block, we measure the prediction error of the whole-block residual. Fig. 3(a) shows distinct low- and high-error regions whose distribution changes with both sampling time and network depth. These results show that residual prediction reliability is itself highly non-uniform across computation locations.

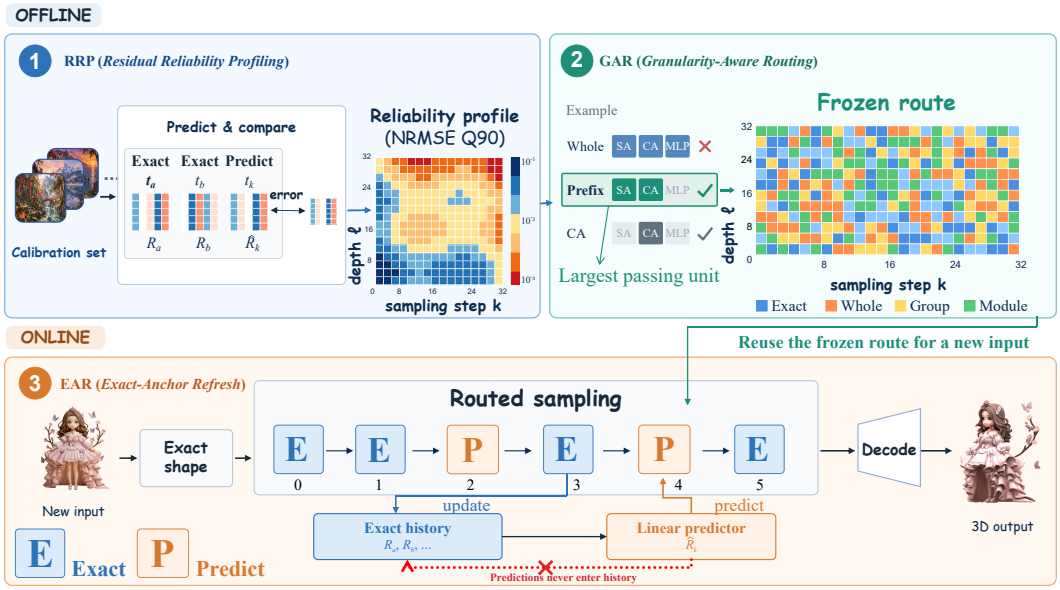

Figure 4: **R3D: profile once, reuse the route across inputs.** *RRP* estimates prediction-error Q90. *GAR* selects the largest-cost reliable residual unit (three candidates shown at one example location). *EAR* maintains two exact anchors: E evaluates exactly and refreshes histories; P follows the frozen route with histories unchanged.

Observation 2: reliability structure is general. We next examine whether the location-dependent reliability structure identified above is shared across different input objects. For each pair of inputs, we compute the Spearman rank correlation between their prediction-error profiles over computation locations. As shown in Fig. 3(b), the error rankings exhibit strong cross-object agreement when all computation locations are considered. Importantly, this agreement remains high when comparisons are restricted to the same sampling step and after removing the shared temporal trend, indicating that the consistency is not explained solely by global temporal variation. Thus, although absolute prediction errors may differ across objects, their relative ordering over computation locations remains largely consistent. Therefore, the reliability information estimated from a calibration set can be generalized to unseen image inputs for image-to-3D synthesis.

Observation 3: predictability is recoverable at finer granularity. We next examine whether finer residual units remain predictable when whole-block prediction is unreliable. Across the evaluated thresholds, finer-unit prediction can reduce the number of residual components requiring exact computation (Fig. 3(c)). This reduction is observed with both per-object oracle selection and a fixed policy derived from a disjoint calibration set. Thus, whole-block prediction failure need not trigger exact computation of the entire block, motivating granularity-aware routing.

To summarize, these observations reveal that residual prediction reliability in image-to-3D generation exhibits a structured and reusable pattern. We leverage this property to selectively replace residual computation with prediction, forming the empirical basis of our proposed **R3D** framework.

3.3 **R3D**: Residual Reliability Routing

Method overview. Our proposed **R3D** consists of three components: (1) *Residual Reliability Profiling (RRP)* estimates residual prediction reliability from a small calibration set. (2) *Granularity-Aware Routing (GAR)* allocates prediction across computation locations and residual granularity according to the resulting reliability profile. (3) *Exact-Anchor Refresh (EAR)* maintains exact histories for temporal residual prediction. **R3D** works during the texture stage of the image-to-3D models to selectively replace residual computations according to a reliable profile while leaving the pretrained model unchanged. The overall framework is illustrated in Fig. 4.

Residual Reliability Profiling (RRP). Observations above show that residual prediction reliability is location-dependent. It also exhibits a consistent cross-input ordering. RRP converts this regular pattern into a reusable reliability profile estimated from a small calibration set $\mathcal{D}_{\text{cal}}$. For compactness, let $c = (k, \ell)$ denote a computation location, and let $R^g_{i,c}$ and $\widehat{R}^g_{i,c}$ denote the exact and predicted residuals of residual unit g for calibration input i, respectively. We measure the normalized prediction error as

$$e^g_{i,c} = \frac{\|\widehat{R}^g_{i,c} - R^g_{i,c}\|_2}{\|R^{\text{whole}}_{i,c}\|_2}, \qquad E^g_c = Q_{0.90}\big(\{e^g_{i,c}\}_{i\in\mathcal{D}_{\text{cal}}}\big)\,. \tag{2}$$

We normalize all residual units by the exact whole-block residual norm, providing a common block-level reference across different residual granularity. The 90th percentile favors residual units whose prediction errors remain small across most calibration inputs rather than only on average. A smaller E^g_c therefore indicates more reliable residual prediction. The collection $\{E^g_c\}$ forms a frozen prediction-error profile that is reused for unseen inputs.

Granularity-Aware Routing (GAR). The observation in Section 3.2 shows that whole-block prediction can fail even when finer residual units remain predictable. In TRELLIS, these finer units correspond to residual updates from the self-attention, cross-attention, and feed-forward submodules within each Sparse Flow Transformer block. Thus **GAR** exploits this granularity-dependent reliability to decide which residual unit should be predicted at each computation location.

For each computation location c, we construct a candidate set $\mathcal{G}_c$ ranging from the whole-block residual to progressively finer residual units. Each candidate $g \in \mathcal{G}_c$ is associated with the calibrated prediction-error score E^g_c produced by RRP. We further denote by $C(g)$ the amount of computation bypassed when g is predicted.

Given a reliability threshold ϵ, GAR selects

$$g^*_c = \underset{g\in\mathcal{G}_c:\, E^g_c<\epsilon}{\arg\max}\; C(g), \tag{3}$$

which chooses the reliable candidate that bypasses the most computation. If no candidate satisfies the reliability criterion, the computation at location c remains exact. **GAR** therefore uses whole-block prediction when it is reliable, while falling back to finer residual units when whole-block prediction is unreliable.

Exact-Anchor Refresh (EAR). Residual prediction relies on valid historical observations. To prevent prediction errors from accumulating over time, **Exact-Anchor Refresh (EAR)** periodically schedules exact evaluations whose residuals are committed as prediction anchors. Predicted residuals are never used as new anchors. Let $\mathcal{E}^g_\ell$ denote the legal scheduled seed/refresh evaluations for residual unit g, and let $\mathcal{A}^g_{k,\ell}$ denote its two most recent committed exact observations before model evaluation k. The anchor history is updated as:

$$\mathcal{A}^g_{k+1,\ell} = \begin{cases} \text{Last}_2\Big(\mathcal{A}^g_{k,\ell} \cup \{(\tau_k, R^g_{k,\ell})\}\Big), & \text{if } k \in \mathcal{E}^g_\ell, \\ \mathcal{A}^g_{k,\ell}, & \text{otherwise.} \end{cases} \tag{4}$$

where $\text{Last}_2(\cdot)$ retains the two most recent exact observations in model time. At a prediction-eligible location outside $\mathcal{E}^g_\ell$, GAR either predicts the selected residual or falls back to exact computation if no candidate is selected. Such fallback exact computation updates the current hidden state but is not committed to the anchor history. Thus, the anchor topology remains fixed and matches that used during calibration.

4 Experiments

We first describe the evaluation protocol in Section 4.1. We then evaluate the quality and efficiency of **R3D** across four image-to-3D backbones in Section 4.2. Finally, Section 4.3 provides ablation studies of the three components of **R3D**: Residual Reliability Profiling (RRP), Granularity-Aware Routing (GAR), and Exact-Anchor Refresh (EAR).

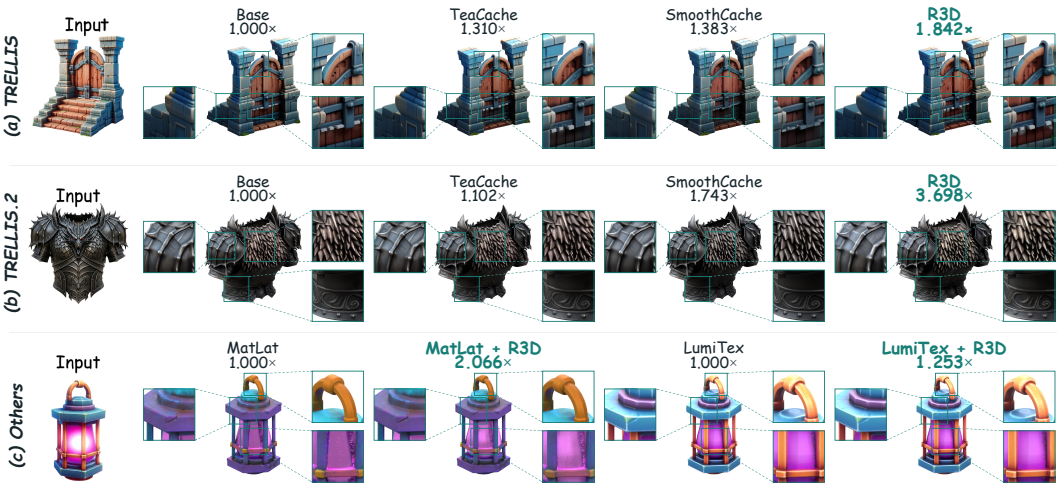

Figure 5: **Appearance across four backbones.** TRELLIS and TRELLIS.2 generate assets from images; MatLat and LumiTex generate materials on supplied geometry. Output views and detail windows are matched within each comparison; input viewpoints differ.

4.1 Experimental Setup

We evaluate the quality and efficiency of **R3D** on **Toys4K-210** (Stojanov et al., 2021), a category-balanced subset containing two objects from each of the 105 Toys4K categories. For each object, all methods use the same conditioning image and are evaluated against the same reference asset under a unified multi-view rendering protocol. In the LumiTex and MatLat transfer experiments, Native and R3D reuse the same untextured mesh generated by TRELLIS.2. We report Chamfer Distance (CD) for geometric fidelity, PSNR and LPIPS (Zhang et al., 2018) for rendered appearance, and Uni3D (Zhou et al., 2023), ULIP-2 (Xue et al., 2024), and CLIP-I (Radford et al., 2021) for image–3D semantic consistency. Efficiency is measured by the wall-clock speedup of the target sampling stage relative to Native execution of the same stage; the reported speedups therefore do not represent end-to-end asset-generation latency. All experiments are conducted on a single NVIDIA H100 GPU. Detailed rendering, metric, and model-specific configurations are provided in the Appendix.

4.2 Main Results

Comparison with training-free acceleration. Table 1 compares **R3D** with Native inference and representative training-free accelerators, TeaCache and SmoothCache, at their reported operating points on TRELLIS and TRELLIS.2. On TRELLIS, **R3D** achieves a $1.842\times$ target-stage speedup while retaining final task-level quality close to Native inference. Relative to Native, PSNR changes by -0.346 dB and LPIPS by $+0.00459$, while CD and the semantic consistency scores remain at similar levels. On TRELLIS.2, **R3D** reaches a $3.698\times$ target-stage speedup, with similarly limited changes in final quality: PSNR changes by -0.179 dB, LPIPS by $+0.00015$, and CD remains nearly unchanged. Across both backbones, the results show that **R3D** can substantially reduce target-stage computation while preserving the final generation quality close to Native inference. Compared with the training-free baselines evaluated in Table 1, this leads to stronger target-stage acceleration under the evaluated settings without a large degradation in task-level quality. Selected outputs and shared detail windows are shown in Fig. 5.

Transfer to other 3D backbones. Beyond the TRELLIS family, we instantiate **R3D** on LumiTex and MatLat using reliability profiles specifically, without updating the pretrained model weights. **R3D** achieves target-stage speedups of $1.253\times$ and $2.066\times$ across the two transfer backbones. The accelerated outputs retain task-level appearance and semantic scores close to their corresponding Native results; for example, LPIPS changes from 0.34575 to 0.34565 on LumiTex and from 0.16697 to 0.16673 on MatLat. We interpret these small metric differences as quality retention rather than improvements over the underlying generators. Overall, the results demonstrate that **R3D** is not

Table 1: **Quantitative comparison with training-free accelerators and our method R3D transfered across diverse backbones on Toys4K-210.** Speedups are measured within each backbone's target sampling stage relative to Native inference and should not be interpreted as end-to-end or cross-backbone runtime comparisons. Quality metrics evaluate rendered appearance, geometry, and image-3D consistency.

		Efficiency	Appearance		Geometry	Semantic Consistency		
Model	Setting	Speedup ↑	PSNR ↑	LPIPS ↓	CD ↓	Uni3D ↑	ULIP-2 ↑	CLIP-I ↑
TRELLIS	Native	1.000×	16.29	0.1949	0.0619	0.3654	0.3956	0.8697
	TeaCache	1.310×	15.90	0.1996	0.0608	0.3676	0.3985	0.8705
	SmoothCache	1.383×	15.95	0.2000	0.0607	0.3670	0.3985	0.8702
	R3D (Ours)	**1.842**×	15.95	0.1995	0.0608	0.3673	0.3980	0.8700
TRELLIS.2	Native	1.000×	18.11	0.2058	0.0602	0.3442	0.3671	0.8418
	TeaCache	1.102×	18.13	0.2062	0.0601	0.3439	0.3691	0.8418
	SmoothCache	1.743×	18.17	0.2056	0.0602	0.3441	0.3678	0.8396
	R3D (Ours)	**3.698**×	18.12	0.2060	0.0602	0.3406	0.3686	0.8383
LumiTex	Native	1.000×	12.71	0.3457	0.0503	0.1117	0.1163	0.7118
	R3D (Ours)	**1.253**×	**12.73**	**0.3456**	0.0503	0.1117	**0.1164**	**0.7119**
MatLat	Native	1.000×	17.07	0.1669	0.0236	0.3094	0.3325	0.8163
	R3D (Ours)	**2.066**×	**17.10**	**0.1667**	0.0236	**0.3102**	**0.3331**	**0.8170**

tied to the TRELLIS architecture and can exploit residual predictability across different backbone families and target-stage formulations.

4.3 Ablation Studies

Ablation studies are conducted on TRELLIS. To isolate the approximation introduced by residual routing, the upstream shape stage is executed exactly, and its resulting support, conditioning features, and initial target-stage state are shared across Native and accelerated runs. The compared variants therefore differ only in the **R3D** configuration applied to the target texture stage. We construct reliability profiles using a disjoint set of 20 calibration objects and freeze the resulting routes before evaluation on a separate 105-object test set; evaluation objects are never used for profiling or route construction. Controlled ablation quality is measured against the paired Native output using PSNR and LPIPS over matched rendered views, together with normalized Chamfer Distance (nCD) between the generated meshes. Detailed configurations and measurement protocols are provided in the Appendix.

Residual Reliability Profiling (RRP). We first test whether the reliability profile provides useful information beyond merely determining the amount of replaced computation. Starting from the calibrated route, we construct three depth-permuted controls that preserve the prediction counts, target costs, and legal prediction masks, while changing the depth locations at which prediction is applied. As shown in Table 2, all variants achieve similar target-stage speedups ($1.214\times$–$1.238\times$), isolating the effect of prediction placement. The profiled route nevertheless yields the lowest LPIPS (5.30×10^{-4} versus 6.35–9.75×10^{-4} for the permuted controls), together with higher PSNR and lower nCD. These controlled results show that RRP captures location-specific predictability that cannot be explained by the prediction budget alone.

Granularity-Aware Routing (GAR). We next isolate the effect of prediction granularity by fixing the H2 refresh topology and varying only the set of admissible residual units. Table 3 shows that restricting prediction to a single granularity limits the amount of computation that can be bypassed. The clearest comparison is between the Submodule and Mixed settings: both select exactly 572 prediction cells, yet GAR replaces 995 residual components instead of 572 by selecting coarser reliable units whenever possible. This increases target-stage speedup from $1.109\times$ to $1.222\times$, while

Table 2: **Ablation of reliability-aware placement.** Depth permutations (Perm.) preserve prediction counts, target costs, and legal masks. LPIPS and nCD are in 10^{-4} and 10^{-3}.

Metric	Profiled placement	Perm. 17	Perm. 43	Perm. 101
ϵ	0.05	0.05	0.05	0.05
Speedup	1.228×	1.214×	1.238×	1.231×
LPIPS ↓	**5.300**	9.752	8.564	6.349
PSNR ↑	**54.227**	50.824	52.518	52.985
nCD ↓	**3.065**	3.116	3.075	3.083

Table 3: **Ablation of routing granularity.** Cells count prediction locations; Units count replaced residual components. $\epsilon = 0.05$. LPIPS and nCD are in 10^{-4} and 10^{-3}.

Metric	Whole	Grouped	Submodule	Mixed (GAR)
Cells	162	261	572	572
Units	486	522	572	**995**
Speedup ↑	1.074×	1.113×	1.109×	**1.222**×
LPIPS ↓	2.81	2.81	5.27	5.30
PSNR ↑	57.28	56.59	54.75	54.23
nCD ↓	3.03	3.02	3.07	3.08

Table 4: **Controlled ablation of Exact-Anchor Refresh.** Quality is measured against paired Native outputs on 105 objects. Shared-GPU timings are exploratory.

History policy	Speedup ↑	LPIPS ↓	PSNR ↑	nCD ↓
EAR (exact anchors only)	1.263×	$\mathbf{5.29\times10^{-4}}$	**54.22**	**0.003074**
Predictions enter history	1.258×	4.00×10^{-3}	42.75	0.003274
Initial exact history frozen	1.290×	0.21659	16.87	0.004944

LPIPS remains nearly unchanged (5.27×10^{-4} versus 5.30×10^{-4}). Thus, the benefit of GAR does not arise simply from issuing more prediction actions; it arises from adapting prediction granularity to the reliability available at each computation location.

Exact-Anchor Refresh (EAR). Finally, we isolate the role of prediction history while holding the compiled route fixed. All variants in Table 4 execute exactly the same 572 prediction actions and replace the same 995 residual components; they differ only in how historical residuals are maintained. Allowing predicted residuals to recursively enter the history produces nearly the same speedup as EAR ($1.258\times$ versus $1.263\times$), but increases LPIPS from 5.29×10^{-4} to 4.00×10^{-3} and decreases PSNR from 54.22 to 42.75 dB. Keeping the initial exact history fixed degrades quality substantially further. These results isolate the two roles of EAR: exact refreshes prevent prediction anchors from becoming stale, while excluding predicted residuals from the history prevents approximation errors from recursively propagating into subsequent predictions.

5 Conclusion

We presented R3D, a training-free acceleration framework for image-to-3D generation and target-stage formulations through residual reliability routing. Our analysis reveals that residual prediction reliability varies systematically across sampling time, network depth, and prediction granularity, while exhibiting consistent patterns across inputs. Based on these observations, R3D combines Residual Reliability Profiling (RRP), Granularity-Aware Routing (GAR), and Exact-Anchor Refresh (EAR) techniques to selectively approximate reliable residuals while retaining exact computation when necessary. Experiments across multiple image-to-3D backbones show that R3D substantially accelerates the target generation stage while preserving geometry, appearance, and semantic quality close to Native inference, without additional training or model modification.

Limitations. Our method **R3D** relies on an offline reliability profile that is tied to a specific model and inference configuration; changes to the checkpoint, sampler, or guidance setting may therefore require re-profiling. This profiling introduces a one-time calibration cost, which is most suitable for repeated deployment where the cost can be amortized. In addition, our controlled generalization study evaluates unseen object instances within the same category universe as calibration, rather than unseen categories. Extending reliability routing to broader distribution shifts is left for future work.

AI USE STATEMENT

We used generative AI tools to assist with literature search, language polishing, manuscript organization, and the formatting and debugging of LaTeX. Generative AI was not used to develop the proposed method, implement experimental code, design experiments, generate or process experimental data, or make conclusions from the experimental results. All AI-assisted text and reference suggestions were reviewed by the authors, and cited works were verified against the original publications. The authors take full responsibility for the accuracy, originality, and integrity of the final manuscript.

REFERENCES

Jingzhi Bao, Hongze Chen, Lingting Zhu, Chenyu Liu, Runze Zhang, Keyang Luo, Zeyu Hu, Weikai Chen, Yingda Yin, Xin Wang, et al. Lumitex: Towards high-fidelity pbr texture generation with illumination context. *arXiv preprint arXiv:2511.19437*, 2025.

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

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. Fastsam3d: 3dfy anything in images but faster, 2026. URL https://arxiv.org/abs/2602.05293.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models, 2020. URL https://arxiv.org/abs/2006.11239.

Yicong Hong, Kai Zhang, Jiuxiang Gu, Sai Bi, Yang Zhou, Difan Liu, Feng Liu, Kalyan Sunkavalli, Trung Bui, and Hao Tan. Lrm: Large reconstruction model for single image to 3d. In B. Kim, Y. Yue, S. Chaudhuri, K. Fragkiadaki, M. Khan, and Y. Sun (eds.), *International Conference on Learning Representations*, volume 2024, pp. 50678–50702, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/file/dcad3425f5c8c36b5b3885c091bf1257-Paper-Conference.pdf.

Hanzhe Hu, Tianwei Yin, Fujun Luan, Yiwei Hu, Hao Tan, Zexiang Xu, Sai Bi, Shubham Tulsiani, and Kai Zhang. Turbo3d: Ultra-fast text-to-3d generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 23668–23678, June 2025.

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.

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.

Yaron Lipman, Ricky T. Q. Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. Flow matching for generative modeling, 2023. URL https://arxiv.org/abs/2210.02747.

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, 2025a. URL https://arxiv.org/abs/2411.19108.

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.

Joseph Liu, Joshua Geddes, Ziyu Guo, Haomiao Jiang, and Mahesh Kumar Nandwana. Smooth cache: A universal inference acceleration technique for diffusion transformers. In *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pp. 3220–3229. IEEE, 2025c.

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

Minghua Liu, Ruoxi Shi, Linghao Chen, Zhuoyang Zhang, Chao Xu, Xinyue Wei, Hansheng Chen, Chong Zeng, Jiayuan Gu, and Hao Su. One-2-3-45++: Fast single image to 3d objects with consistent multi-view generation and 3d diffusion. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 10072–10083, June 2024.

Xiaoxiao Long, Yuan-Chen Guo, Cheng Lin, Yuan Liu, Zhiyang Dou, Lingjie Liu, Yuexin Ma, Song-Hai Zhang, Marc Habermann, Christian Theobalt, and Wenping Wang. Wonder3d: Single image to 3d using cross-domain diffusion. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 9970–9980, June 2024.

Cheng Lu, Yuhao Zhou, Fan Bao, Jianfei Chen, Chongxuan Li, and Jun Zhu. Dpm-solver: A fast ode solver for diffusion probabilistic model sampling in around 10 steps. *Advances in neural information processing systems*, 35:5775–5787, 2022.

Lars Mescheder, Michael Oechsle, Michael Niemeyer, Sebastian Nowozin, and Andreas Geiger. Occupancy networks: Learning 3d reconstruction in function space. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

William Peebles and Saining Xie. Scalable diffusion models with transformers, 2023. URL https://arxiv.org/abs/2212.09748.

Ben Poole, Ajay Jain, Jonathan T. Barron, and Ben Mildenhall. Dreamfusion: Text-to-3d using 2d diffusion. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=FjNys5c7VyY.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision. In Marina Meila and Tong Zhang (eds.), *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pp. 8748–8763. PMLR, 18–24 Jul 2021. URL https://proceedings.mlr.press/v139/radford21a.html.

Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. *arXiv preprint arXiv:2010.02502*, 2020.

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.

Nanyang Wang, Yinda Zhang, Zhuwen Li, Yanwei Fu, Wei Liu, and Yu-Gang Jiang. Pixel2mesh: Generating 3d mesh models from single rgb images. In *Proceedings of the European Conference on Computer Vision (ECCV)*, September 2018.

Jianfeng Xiang, Xiaoxue Chen, Sicheng Xu, Ruicheng Wang, Zelong Lv, Yu Deng, Hongyuan Zhu, Yue Dong, Hao Zhao, Nicholas Jing Yuan, and Jiaolong Yang. Native and compact structured latents for 3d generation. *Tech report*, 2025a.

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, 2025b. URL https://arxiv.org/abs/2412.01506.

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, 2024. URL https://arxiv.org/abs/2305.08275.

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 *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 21481–21491. IEEE, 2025.

Kyeongmin Yeo, Yunhong Min, Jaihoon Kim, and Minhyuk Sung. Matlat: Material latent space for pbr texture generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2026.

Longwen Zhang, Ziyu Wang, Qixuan Zhang, Qiwei Qiu, Anqi Pang, Haoran Jiang, Wei Yang, Lan Xu, and Jingyi Yu. Clay: A controllable large-scale generative model for creating high-quality 3d assets, 2024. URL https://arxiv.org/abs/2406.13897.

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.

Junsheng Zhou, Jinsheng Wang, Baorui Ma, Yu-Shen Liu, Tiejun Huang, and Xinlong Wang. Uni3d: Exploring unified 3d representation at scale, 2023. URL https://arxiv.org/abs/2310.06773.

A STATEMENTS

A.1 LIMITATIONS

A frozen reliability profile can become inappropriate after a change in checkpoint, sampling schedule, guidance, input distribution, or representation support. Calibration also incurs a one-time cost that may not be amortized in small deployments. The evaluation sets are held out from calibration and development, but the controlled split tests new object identities within the same category universe. Generalization to unseen categories or substantially different input distributions remains unverified.

Fine residual targets may remain locally predictable when a whole block fails, but local prediction error and final perceptual quality are different quantities. Long extrapolation horizons can reduce the set of candidates passing calibration, and permissive thresholds can increase final error. The direct-reuse comparison also shows that linear extrapolation does not dominate every metric at every threshold. Likewise, the tested online guard is one implementation; its overhead does not imply that all causal online adaptation must be slower than a frozen route.

Quality-tail examples and EAR distributions expose difficult cases without establishing equivalence of output distributions. The shared-GPU EAR timings remain exploratory. Evaluation on broader input distributions, lower profiling cost, and low-overhead adaptation remain useful extensions.

A.2 ETHICAL STATEMENT

R3D accelerates existing pretrained 3D generators without updating their weights. It does not address biases, privacy risks, or content-safety limitations inherited from these generators. Reducing inference cost may also make inappropriate uses easier to scale. Use of the method should respect the licenses and access conditions of the underlying models and datasets, as well as the rights associated with input images and generated assets. The acceleration mechanism provides no additional content-safety guarantee.

A.3 REPRODUCIBILITY STATEMENT

Appendix B describes the datasets, held-out evaluation protocol, generation configurations, baseline adaptations, timing scopes, and evaluation metrics. Appendix C.1 specifies the residual targets, route compiler, anchor calendar, and runtime history rules. Appendices C.3 and C.4 distinguish controlled mechanism studies from model-specific deployment configurations. Reproduction requires the fixed input manifests, checkpoint and sampler configurations, calibration identities, serialized routes, and paired generation fixtures. Appendix B.7 details these artifacts and the validation procedure. Evaluation objects are not used for calibration, development, or operating-point selection, and model weights remain unchanged.

B EXPERIMENTAL SETTINGS

B.1 DATASETS AND EVALUATION SPLITS

Task-level evaluation. The task benchmark uses 210 Toys4K (Stojanov et al., 2021) objects, two from each of 105 categories. The asset list follows the first two eligible Blender assets per category in the archived lexicographic selection. All configurations use this fixed input list and compare against the corresponding reference assets. This benchmark evaluates final task quality, whereas the controlled study below measures deviation from a paired full-compute output. The two reference conventions must not be conflated.

Controlled TRELLIS evaluation. The controlled set contains a further 105 objects, one per category, drawn from the separate group of third eligible assets. The frozen manifest, rather than a numeric filename suffix, defines membership. For example, gaps in available asset identifiers mean that the third eligible object need not have suffix `002`. These 105 objects are disjoint from the 20 route-calibration and 12 development objects by identity and input-image hash. The calibration,

development, and evaluation sets share the same category universe; this is an object-level, not an unseen-category, comparison.

Calibration and development. Route calibration uses 20 objects, and development uses a separate set of 12 objects. Both sets are disjoint from the 210-object task benchmark and the 105-object controlled evaluation set. Object identities and input-image hashes are specified by the fixed split manifests. Calibration constructs residual-error profiles and routes without updating pretrained weights; development supports implementation and protocol checks. Neither evaluation set is used for these purposes.

Evaluation isolation. The evaluation sets remain untouched during calibration and development. Evaluation inputs, residual diagnostics, and output metrics are not used to construct profiles, select thresholds or routes, or revise the implementation before evaluation. Configurations are fixed before the corresponding evaluation runs. Subsequent diagnostics and qualitative selections describe the evaluated outputs and do not feed back into those configurations.

Table 5: **Data roles and comparison units.** Counts refer to objects, not views or timing repetitions. The 20/12 split belongs to the controlled TRELLIS study; each additional-model diagnostic has its own 10/30 split.

Data role	Objects	Use and relationship
Task benchmark	210	Untouched task evaluation against reference assets; separate from calibration and development.
Route calibration	20	Profile construction only; disjoint from development and evaluation objects.
Development	12	Implementation/protocol checks; disjoint from calibration and evaluation objects.
Controlled evaluation	105	Untouched evaluation against paired Native outputs; disjoint from the 20/12 objects.
Other-model probes	10/30	Separate calibration/untouched evaluation sets within each model; distinct from deployment.

B.2 Generation configuration and paired inputs

The controlled study uses TRELLIS-image-large with 24 target-stage Transformer blocks and 25 model evaluations. The sampler is Flow-Euler-Guidance-Interval-Sampler with $\sigma_{\min} = 10^{-5}$, guidance strength 5, guidance interval $[0.5, 1.0]$, and time rescaling 3. Model arithmetic uses FP16 and diagnostic squared-norm reductions use FP32. Conditional and negative branches follow the native execution mask; an absent branch is not an exact observation or a zero-error cell.

For each controlled object, shape stage is executed exactly and its sparse support, image-conditioning features, and initial texture-stage state are stored as a fixture. All compared routes replay that fixture. shape stage uses seed 0; texture stage noise continues the recorded CPU random-number stream rather than being assigned a newly invented independent seed. This pairing isolates changes inside the target sampler. It does not imply that decoded geometry is fixed: TRELLIS SLat features affect both appearance and the final decoded surface even when the upstream support is unchanged.

The deployment experiments use model-specific configurations. For each input in the LumiTex and MatLat transfer experiments, Native and R3D reuse the same untextured mesh generated by TRELLIS.2, keeping the input geometry fixed. Their outputs are rendered on GPU using the same evaluation settings. The controlled TRELLIS study separately replays the sparse support, conditioning features, and initial texture stage state described above. Model-specific deployment details are given in Appendix C.4.

B.3 Baseline adaptations

Native. Native executes the target model without residual approximation. A baseline's timing ratio uses the Native arm measured in its own batch. Independently measured Native latencies need

not be identical across batches; nominally equal target-stage scopes alone do not establish identical machine load or software overhead.

TeaCache. Our TeaCache (Liu et al., 2025a) adapters reuse the residual of the complete target Transformer stack. The probe is the first block's modulated pre-attention input. Its relative L_1 change is mapped by a backbone-specific fourth-degree polynomial fitted to exact trajectories; the fit target is the change of the stack's output embedding. This fitted target is distinct from the cached stack residual. At a cache hit, the saved stack residual is added to the current stack input, while input/output processing and sampler updates still execute. The adapters preserve native classifier-free guidance and maintain branch-specific state. First/last steps and incompatible cache states require exact computation. Negative polynomial estimates are clipped to zero; the TRELLIS adapter additionally rejects probes outside its recorded calibration range. The same range-rejection rule is not assumed for the separate TRELLIS.2 adapter.

SmoothCache. The SmoothCache (Liu et al., 2025c) adapters measure relative changes in SA, CA, and MLP outputs on exact trajectories, aggregate the calibration profiles, and compile static reuse schedules. The scanner chooses the longest admissible lookahead endpoint under the configured error threshold, with exact first and last evaluations. Runtime reuse substitutes cached submodule outputs while retaining current normalization, modulation, gates, and residual addition. This differs from R3D's prediction of endpoint residual updates. Guidance histories and schedules are separated by branch, and each sampling trajectory starts with fresh caches.

Table 6: **Recorded cache-baseline operating points.** Thresholds belong to different baseline criteria and are not interchangeable with R3D's ϵ. These are the operating points used in the main deployment comparison.

Backbone	Baseline	Threshold	Additional setting
TRELLIS	TeaCache	0.1	Backbone-specific degree-4 fit
TRELLIS	SmoothCache	0.1	Lookahead 3
TRELLIS.2	TeaCache	0.1	Material-1024 degree-4 fit
TRELLIS.2	SmoothCache	0.3	Lookahead 3

Operating-point sensitivity. The main comparison reports one operating point for each cache baseline. To characterize the corresponding efficiency–fidelity trade-off, we additionally sweep the operating thresholds of TeaCache and SmoothCache over $\{0.1, 0.2, 0.3, 0.5, 0.8\}$ on TRELLIS and TRELLIS.2. For each setting, we report the recorded target-stage speedup together with LPIPS deviation from the corresponding Native output.

Figs. 6 and 7 show that increasing the aggressiveness of the cache baselines can extend their acceleration, but generally at the cost of larger deviation from Native inference. On TRELLIS, R3D remains in the low-deviation regime at its reported operating point relative to the neighboring TeaCache settings. On TRELLIS.2, TeaCache can be pushed to nearly the same acceleration regime as R3D, but incurs larger paired-Native LPIPS, while SmoothCache remains in a lower-speed regime over the evaluated sweep. Because the baseline trade-offs are not strictly monotonic with threshold, we report all evaluated operating points rather than selecting individual favorable settings.

B.4 Timing protocol and aggregation

Controlled TRELLIS timings use one warmup and four CUDA-synchronized repetitions per arm, with interleaved execution order. For per-object median latencies t_i^N and t_i^M, the dataset-level speedup is

$$S_M = \frac{\sum_{i=1}^{N} t_i^N}{\sum_{i=1}^{N} t_i^M}. \tag{5}$$

This ratio is not the arithmetic mean of individual speedups. Routing, residual prediction, history access, exact refresh, and sampler computation inside the timed Stage 2 call are included. Input preparation, Stage 1, decoding, mesh export, and rendering are excluded. In the controlled route experiments, controller attachment and removal occur outside this window. The later TeaCache/SmoothCache deployment batches include controller construction/reset/attachment/removal

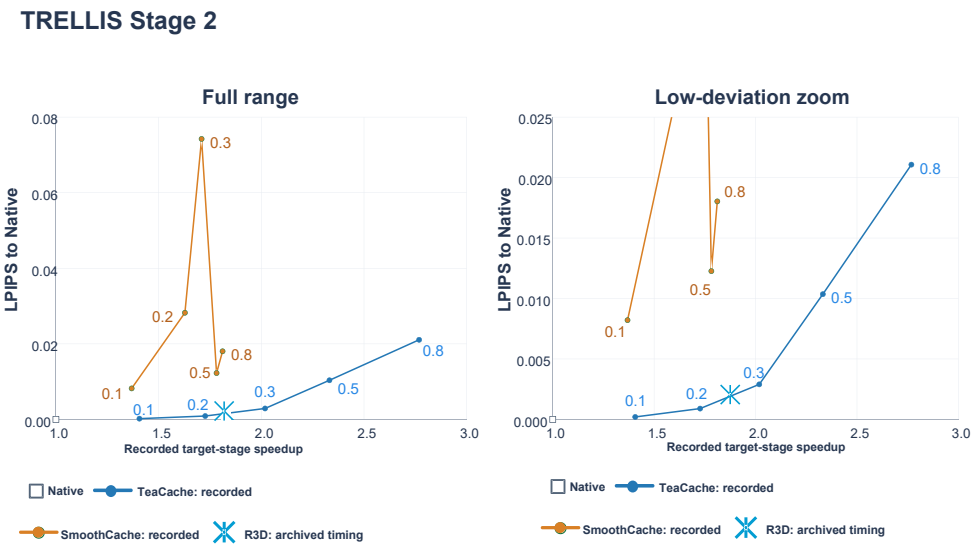

Figure 6: **Baseline operating-point sweep on TRELLIS.** Target-stage speedup versus paired-Native LPIPS for TeaCache and SmoothCache under different operating thresholds. R3D is shown at the operating point used in the main deployment comparison. The right panel enlarges the low-deviation region.

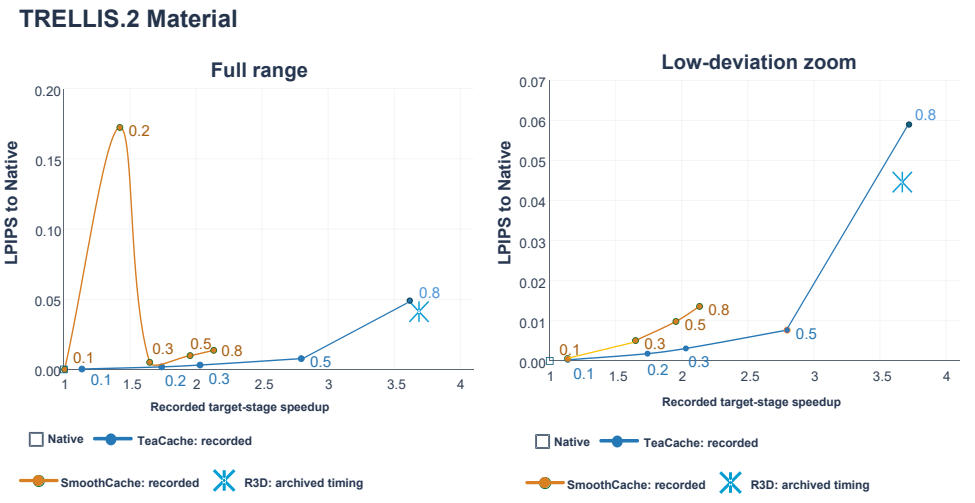

Figure 7: **Baseline operating-point sweep on TRELLIS.2.** More aggressive TeaCache settings approach the target-stage acceleration of R3D but incur larger paired-Native LPIPS, whereas SmoothCache remains in a lower-speed regime over the evaluated threshold range. The right panel enlarges the low-deviation region.

within their resident-sampler timing window. We retain the recorded scopes instead of treating these small implementation-boundary differences as a fully matched overhead comparison.

Runs use an NVIDIA H100 80GB HBM3 GPU. The audited controlled environment records PyTorch 2.6.0+cu124, CUDA 12.4, flash-attn 2.7.3, and spconv-cu120 2.3.6, with TF32 disabled. Repeated timings estimate latency for each object and are not additional independent objects. The dedicated EAR rerun enables deterministic algorithms for all four arms and records shared-GPU permission; its timing interpretation is specified in Appendix C.3.5.

Deployment scopes are TRELLIS texture stage, TRELLIS.2 material flow, LumiTex AL-latent sampling, and MatLat material-latent generation. They cover different portions of their respective pipelines and are not end-to-end asset-generation latencies. The independent pipeline-time breakdown in Fig. 24 uses a separate 105-input measurement through decoding; it is not the denominator of the 210-input target-stage speedups.

Shape-stage motivation renders. Fig. 2 re-renders four archived Fast3Dcache/Native comparison pairs from shape-stage-only acceleration runs. The archived meshes and their original materials are unchanged. Each pair uses a common bounding volume, orthographic camera, lighting rig, and 3072-pixel output; display crops and local windows are shared within the pair. These are new views of existing outputs, not regenerated inference runs. Matching input/seed and exact texture-stage execution follow the original author-confirmed protocol; the new render audit does not independently reconstruct the original sampling records. The cases illustrate possible structural differences under shape-stage acceleration rather than a failure frequency or a measured per-case effect size. No speedup is assigned to these individual pairs.

B.5 Rendering, quality metrics, and statistical units

Paired approximation error. Each controlled output is compared with its paired Native output over 24 matched views. LPIPS and PSNR are first averaged over views within an object, then over objects. LPIPS uses the AlexNet v0.1 network (Zhang et al., 2018). The paired geometry metric samples 20,000 area-weighted surface points per mesh. For candidate points P, Native points Q, and the Native bounding-box diagonal d_N, it is

$$\mathrm{nCD}(P,Q) = \frac{1}{2d_N}\left(\frac{1}{|P|}\sum_{p\in P}\min_{q\in Q}\|p-q\|_2 + \frac{1}{|Q|}\sum_{q\in Q}\min_{p\in P}\|q-p\|_2\right). \tag{6}$$

Distances are unsquared. Independent finite point samples introduce a sampling contribution, so mesh-distance magnitudes should not be interpreted as exact analytic surface distances. Tables state their scaling explicitly: most controlled tables use 10^{-4} for LPIPS and 10^{-3} for nCD; the dedicated EAR table below reports LPIPS in raw units and nCD in units of 10^{-3}.

Task-level quality. For the 210-input benchmark, PSNR/LPIPS compare generated renders with reference renders. The recorded protocol uses 12 views, 512×512 resolution, 32 rendering samples per pixel, and white backgrounds. Aligned CD compares generated and reference geometry after normalization and the recorded rigid-alignment procedure; it is distinct from the Native-diagonal-normalized paired nCD above. Uni3D (Zhou et al., 2023), ULIP-2 (Xue et al., 2024), and CLIP-I (Radford et al., 2021) compare the generated result with the conditioning image. Apparent small improvements over Native in these task metrics do not establish that approximation improves the generator. For LumiTex and MatLat, Native and R3D are evaluated using the same input geometry and matched GPU rendering settings.

Uncertainty. Objects are the resampling unit. For a paired contrast, the same sampled object indices are used for both methods; for speedup, both latency sums in Eq. (5) are recomputed. We do not treat the 24 views, timing repetitions, or all object pairs as independent samples. TRELLIS reliability/recovery diagnostics use 2,000 object-bootstrap replicates; the additional-generator diagnostics use 1,000, and the dedicated EAR comparison uses 10,000 with seed 20260908. The corresponding intervals condition on the selected calibration objects and frozen routes. They do not include calibration-selection uncertainty, and pointwise 95% intervals are neither multiplicity-adjusted tests nor equivalence guarantees.

B.6 Offline profiling cost

For the primary TRELLIS Mixed-H2 route at $\epsilon = 0.05$, collecting 20 calibration objects takes 1103.71 s and compiling the route takes 10.11 s: 1113.82 s in total. In the corresponding overhead experiment, Native and accelerated mean texture stage times are 2.83115 s and 2.30544 s. If the route is reused under comparable conditions, the projected break-even count is

$$n_{\text{amort}} = \left\lceil \frac{1113.82}{2.83115 - 2.30544} \right\rceil = 2119. \tag{7}$$

This is an amortization projection from measured costs, not an experiment on 2,119 additional objects. It excludes upstream generation and decoding, and does not combine the latency of another deployment or EAR rerun with this profiling cost.

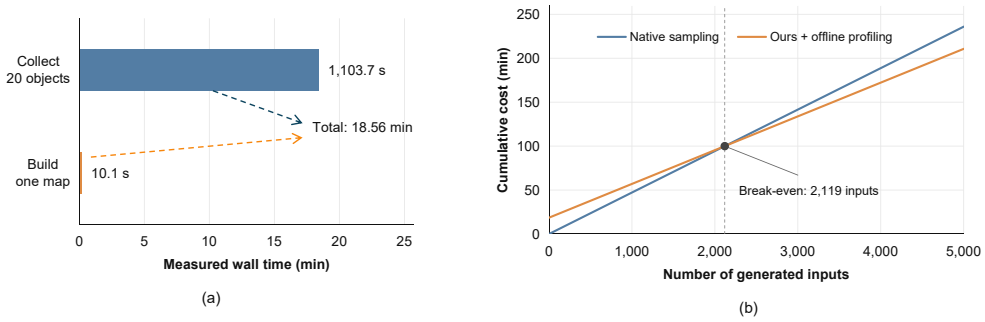

Figure 8: **Profiling cost and projected amortization.** The measured one-time collection/compilation cost is amortized using the paired Stage 2 latency saving from the same overhead study.

B.7 Reproduction artifacts and validation

Reproduction requires the object manifest and input hashes; checkpoint and model configuration; sampler times and guidance mask; residual definitions and normalization; calibration identities and aggregation; anchor calendar; tie-breaking; and serialized prediction actions. The matched-history route has 572 selected cells and 995 component units. Recorded hashes distinguish the manifest, route, code, weights, and evaluator.

Replay the shared support, conditioning, and initial noise, and check the exact controller path against Native. Quality and timing runs must use identical numerical settings, with output validation outside the timed window. Use matched cameras and aggregate views within objects. A new model or sampler configuration requires compatible calibration profiles.

C Supplementary Results and Analysis

C.1 Route Construction and Runtime Implementation

C.1.1 Residual endpoints, branch identity, and normalization

For a serial SA–CA–MLP block, write $h^{(0)}$ for the block input and $h^{(1)}, h^{(2)}, h^{(3)}$ for the outputs after the three residual sublayers. These states include the model's native modulation, gates, and residual additions. A grouped residual is an endpoint difference on the exact trajectory, rather than the sum of separately predicted module outputs. At inference, a predicted endpoint difference is added to the current input of that group; surrounding modules operate on the resulting current state.

Table 7: **TRELLIS candidate units.** The cost proxy counts bypassed components; it is not measured FLOPs or latency. Candidates overlap, so at most one unit is selected per branch/time/block location.

Unit	Exact residual	Remaining exact modules	$C(g)$
Whole	$h^{(3)} - h^{(0)}$	None	3
Prefix	$h^{(2)} - h^{(0)}$	MLP	2
Suffix	$h^{(3)} - h^{(1)}$	SA	2
SA	$h^{(1)} - h^{(0)}$	CA, MLP	1
CA	$h^{(2)} - h^{(1)}$	SA, MLP	1
MLP	$h^{(3)} - h^{(2)}$	SA, CA	1

We expand the main-paper location notation to $c = (b, k, \ell)$ when guidance branch b must be explicit. Histories are keyed by branch, block, and residual target. An exact CA observation is not a Whole anchor, nor can a conditional history be used by the negative branch. Predictor coefficients use the actual model times τ_k in Eq. (1); uniform evaluation-index spacing is not substituted for a rescaled sampler's time spacing.

The controlled route uses direct residual error normalized by the exact Whole residual norm, as in Eq. (2). Invalid or nonfinite observations and zero denominators do not certify a candidate as reliable. A valid profile requires the designated calibration observations and compatible two-anchor histories. The recorded compiler computes the sample Q90 with linear interpolation and uses the strict test $E_c^g < \epsilon$. This is an empirical calibration criterion, not a probabilistic upper bound on future rollout error.

The dense whole-block diagnostics and the schedule-matched routing profiles answer different questions. Dense diagnostics measure local predictability on exact trajectories while continuously updating exact observations. Deployable H1/H2/H3 routes instead use only their scheduled anchors. No dense diagnostic map is silently reused as a guarantee for a longer extrapolation horizon.

C.1.2 Compiling a frozen route

The Whole family permits only Whole; Grouped permits Prefix and Suffix; Submodule permits SA, CA, and MLP; and Mixed permits all six. At a prediction-eligible location, GAR chooses the passing candidate with largest $C(g)$. Equal-cost ties use the fixed order Whole, Prefix, Suffix, SA, CA, MLP, restricted to the active family. If none passes, the location remains exact. This local allocation uses the component proxy; it does not claim globally optimal wall-clock scheduling.

The schedule is compiled together with its seed/refresh calendar and legal native branch mask. An action is valid only when the required earlier scheduled exact observations exist. The serialized route records predictor, thresholds, calibration identities, target choices, histories, and configuration provenance. Consequently, applying a route to a different sampler schedule, guidance mask, checkpoint, or candidate definition requires a compatibility check and generally re-profiling.

C.1.3 Exact anchors, refresh schedules, and reset conditions

For a continuously active branch, zero-indexed evaluations 0 and 1 seed the histories. Thereafter Hr permits r prediction-eligible evaluations between exact refreshes. The calendar repeats predict slots followed by a refresh; a legal mask also checks whether the branch actually executes. For illustration, the first nine roles are

Evaluation k	0	1	2	3	4	5	6	7	8
H1	E	E	P	E	P	E	P	E	P
H2	E	E	P	P	E	P	P	E	P
H3	E	E	P	P	P	E	P	P	P

Here E is a scheduled exact anchor and P is eligible for prediction, not a promise that every block will be skipped. Exact computation at a rejected P slot does not update the histories. This keeps the runtime anchor topology matched to calibration; otherwise a route would use an unprofiled, input-dependent calendar.

Table 8: **Offline compilation and online execution.** This procedure describes the frozen controlled route. All calibration rollouts are exact; deployment uses input-specific histories.

	Offline: RRP and GAR
1	Fix checkpoint, sampler, guidance mask, candidate family, threshold, and anchor calendar.
2	Execute each calibration input exactly. Record target residuals; update diagnostic histories only at calendar seed/refresh evaluations.
3	At each legal prediction slot, extrapolate from the two scheduled exact anchors and measure the direct residual error for each candidate. Do not inject the prediction into the calibration trajectory.
4	Aggregate the per-input errors into E_c^g. Among valid candidates with $E_c^g < \epsilon$, choose the maximum-cost unit with deterministic tie-breaking; otherwise mark exact execution.
5	Validate branch masks and anchor availability; freeze the route and its configuration.
	Online: frozen execution with EAR
6	Initialize the new input's target-stage state and empty target-specific histories.
7	At seed/refresh slots, execute exactly and retain the latest two scheduled exact residuals and their model times for each required target.
8	At a prediction slot with a selected target, evaluate Eq. (1), bypass that target, and execute surrounding modules on the current state. Do not commit the prediction to history.
9	At an unselected prediction slot, execute exactly without advancing the frozen anchor calendar. Continue the native sampler update.

Histories belong to a single input trajectory and compatible sparse support, conditioning, target definition, and branch identity. A new trajectory starts with empty histories. Incompatible histories must not be reused. The controlled implementation validates these requirements and rejects illegal/missing-anchor actions rather than silently replacing them with another cache policy. EAR's exactness is local: refreshes exactly evaluate the model on the current, possibly already approximated hidden state. They do not restore the trajectory to the hypothetical full Native trajectory, so EAR does not prove that all accumulated state error vanishes.

C.1.4 Route composition and resource implications

The primary Mixed-H2 route at $\epsilon = 0.05$ has 572 prediction actions among 1,056 native branch/block calls. Its selected units are 162 Whole, 94 Prefix, 5 Suffix, 11 SA, 300 CA, and no MLP-only actions. Their component proxy is

$$162 \cdot 3 + (94 + 5) \cdot 2 + (11 + 300) \cdot 1 = 995. \tag{8}$$

Thus 572 is a count of selected locations and 995 is a count of bypassed components; neither is a percentage or a FLOP count. By module, the route bypasses 267 SA, 561 CA, and 167 MLP calls. The exact remaining calls are 789, 495, and 889, respectively. Prediction and history operations still incur work, which is why the component proxy is not itself a speedup estimator.

Each maintained history stores two residual tensors plus model-time metadata. If a target residual has n elements stored with b bytes per element, its anchor storage is $2nb$ bytes before container overhead. Histories can be limited to the targets required by the compiled route. The routing map has no per-input fitting step, but tensor size, sparse-token count, memory traffic, and Python/controller work still affect realized latency. We report measured latency rather than inferring it from the number of skipped operations.

C.2 Additional Evidence for Structured Residual Reliability

C.2.1 Cross-object consistency beyond a shared temporal trend

The TRELLIS diagnostic records whole-block prediction errors along exact trajectories for the same 105 evaluation objects used by the controlled study. Warmup and unavailable locations are excluded rather than assigned zero error. We report three complementary statistics: (i) pooled-grid correlation over the common time–depth grid, (ii) fixed-time correlation across depth within each sampling evaluation, and (iii) detrended correlation after subtracting each object's per-step mean log error. The median pairwise Spearman correlations are:

Branch	Pooled grid	Fixed time	Detrended
Conditional	0.9052	0.8809	0.8592
Negative	0.9288	0.8804	0.8634

Let $x_i(k, \ell)$ be object i's log whole-block error on the common valid grid. The pooled statistic ranks all available time–depth cells. For fixed-time agreement, we first compute object-pair correlations across depth at each evaluation, take the median across pairs, and then the median across defined evaluations, giving each evaluation equal weight. Detrending replaces $x_i(k, \ell)$ by

$$x_i(k, \ell) - \frac{1}{|\mathcal{L}_k|} \sum_{m \in \mathcal{L}_k} x_i(k, m), \tag{9}$$

before ranking over the grid. This removes each object's depth-averaged temporal trend. The 105 objects yield 5,460 unordered pairs, but the pairs are dependent because they share objects. Bootstrap replicates therefore resample object identities and recompute the aggregate; they do not resample 5,460 pairs as independent observations. The dense diagnostic uses exact trajectories without prediction injection.

The persistence of high agreement under fixed-time and detrended comparisons indicates that the shared ordering is not explained only by an easy early/late sampling trend. Object-pair correlations are dependent; where uncertainty is reported, objects rather than individual pairs or cells are resampled.

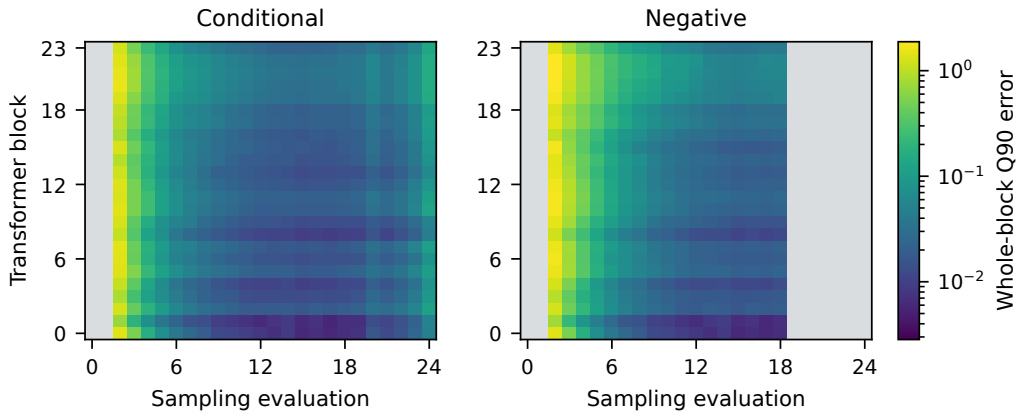

Figure 9: **Whole-block reliability maps for both TRELLIS guidance branches.** Gray cells are unavailable under the native execution/history mask.

C.2.2 How much structure is explained by time and depth?

We further test whether the whole-block reliability map can be summarized by low-complexity time and depth effects. For each guidance branch, we fit time-only, depth-only, additive, and unrestricted time–depth profiles to the calibration-set mean log prediction error. All profiles are frozen before evaluation. Table 9 evaluates their agreement with the 105 evaluation objects. The additive profile captures most of the ordering, while the unrestricted profile provides a smaller but consistent improvement. For example, conditional mean Spearman agreement is 0.9364 for the additive profile and 0.9424 for the unrestricted profile, compared with 0.7558 for time alone and 0.4585 for depth alone. This supports structured reliability while also showing that much of its ordering has a compact explanation; it does not establish that arbitrary time–depth interactions are always necessary.

C.2.3 Finer-grained opportunities after whole-block failure

To complement Fig. 3(c), we quantify the fraction of whole-block computation that can be recovered by a calibration-frozen finer-grained choice. Let F_i be locations where whole-block prediction fails

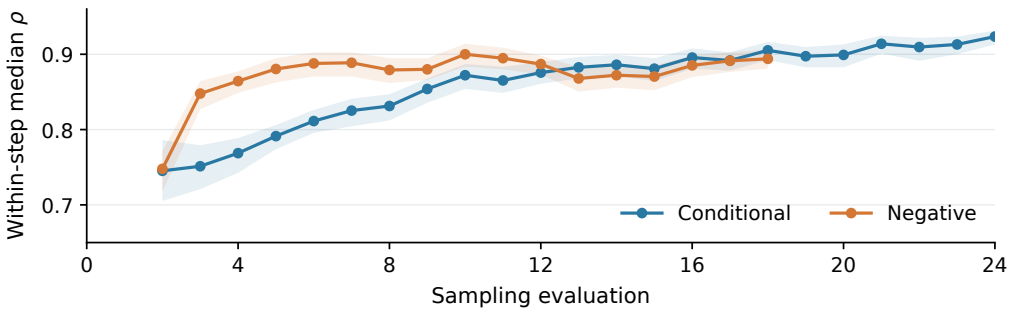

Figure 10: **Cross-object reliability agreement at individual sampling evaluations.** Correlations are computed across depth within each evaluation, complementing the pooled statistic in the main paper.

Table 9: **A compact description of whole-block error ordering.** Fits use mean log self-normalized error from 20 calibration objects under dense exact-observation diagnostics. Frozen reliability profiles are evaluated on 105 calibration-disjoint evaluation objects. Spearman and log-MAE are averaged over objects; intervals are 95% object-bootstrap intervals with calibration fixed. This characterizes reliability profiles, separately from approximate-sampling routes.

Branch	Fitted profile	Mean ρ	95% interval	Log-MAE
Conditional	Time only	0.7558	[0.7424, 0.7681]	0.4112
	Depth only	0.4585	[0.4470, 0.4698]	0.7952
	Additive	0.9364	[0.9293, 0.9427]	0.2750
	Unrestricted	0.9424	[0.9355, 0.9487]	0.2600
Negative	Time only	0.7545	[0.7399, 0.7686]	0.5078
	Depth only	0.4649	[0.4538, 0.4761]	1.0245
	Additive	0.9509	[0.9439, 0.9570]	0.3521
	Unrestricted	0.9538	[0.9470, 0.9598]	0.3395

for evaluation object i. The recovered cost fraction is

$$W_i = \frac{\sum_{j \in F_i} C(g_j)\, \mathbf{1}[g_j \text{ passes on } i]}{\sum_{j \in F_i} C(\text{Whole})}, \tag{10}$$

where g_j is selected using calibration data only. A location contributes zero if no frozen finer target passes. At most one target is credited per location, avoiding double counting overlapping Prefix/Suffix/submodule opportunities. This diagnostic excludes standalone negative-branch CA to avoid counting that special case as evidence for a general finer-unit opportunity; it does not imply that the deployed Mixed route excludes negative CA. The oracle version allows an object-specific finer target and is reported only as an upper diagnostic reference. Confidence intervals resample evaluation objects while keeping calibration fixed.

At $\epsilon = 0.05$, object-specific opportunity is 22.70%, whereas calibration-frozen recovery is 12.44% (95% interval 12.09–12.82%). The gap distinguishes knowing an evaluation object's errors from making an input-independent choice. The frozen recovery percentage is not monotone in ϵ: both the selected candidates and the whole-failure denominator change as the threshold changes. It is a local cost-weighted diagnostic, not a measurement of final output quality or realized acceleration.

C.2.4 Residual-reliability diagnostics on additional generators

We repeat the exact-observation reliability analysis on TRELLIS.2, LumiTex, and MatLat. Each uses disjoint sets of 10 calibration and 30 evaluation objects, separately from the 210-input deployment study. The 30-object evaluation set remains untouched until the model-specific diagnostic configuration is fixed. TRELLIS.2 probes material flow, LumiTex the joint AL-token stream, and

Table 10: **Component-count-weighted finer-unit recovery on TRELLIS.** Twenty calibration objects select finer targets, evaluated on 105 objects under H2 with common whole-block normalization. Recovered component count is divided by the whole-block component count over the same whole-failure cells. SA, CA, and MLP each contribute one component to the proxy; prediction overhead is not subtracted. Standalone negative-branch CA is excluded. Oracle opportunities allow per-object target choice; finer-unit recovery with frozen choices does not. Intervals use 2,000 object-bootstrap samples with the calibration-selected targets fixed.

ϵ	Oracle opportunity (%)	Finer-unit recovery (%)	Frozen 95% CI
0.02	17.51	10.09	[9.94, 10.26]
0.05	22.70	12.44	[12.09, 12.82]
0.10	21.95	11.14	[10.73, 11.53]

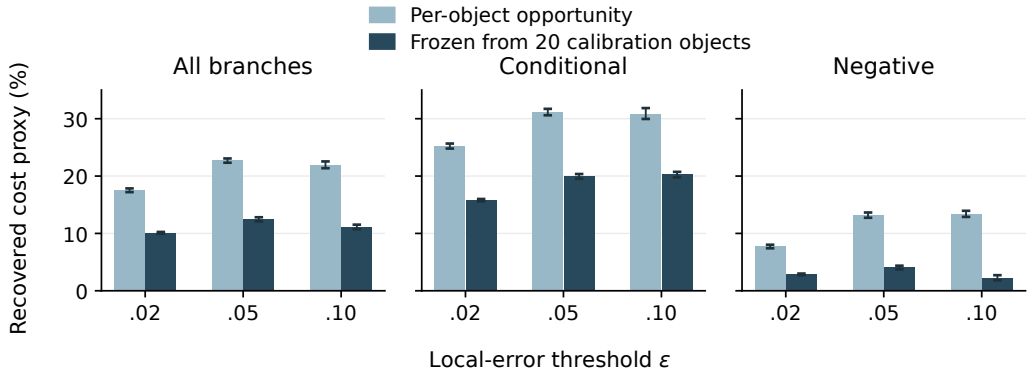

Figure 11: **Branch-specific finer-unit recovery.** Same H2 diagnostic as Fig. 3(c), separated by guidance branch; component-cost fractions do not measure runtime savings.

MatLat its material DiT. These probes are intentionally within-model: normalized time/depth axes align visualization extents but do not imply correspondence between physical layers or sampler times across architectures. Table 11 reports cross-input whole-block ordering and finer-unit recovery for each model/branch. The corresponding held-out whole-block reliability maps are shown in Figs. 12–15.

Table 11: **Residual-predictability structure in additional 3D generators.** Each model uses 10 calibration and 30 untouched, calibration-disjoint evaluation objects. Correlations summarize whole-block error orderings; fixed-time correlations compare depth within each time. Recovery uses self-normalized error at $\epsilon = 0.05$ and calibration-selected finer-unit targets. These are within-model diagnostics, not directly comparable runtime savings; Fig. 3 uses a different normalization and component-count weighting.

Model	Branch	All-cell ρ	Fixed-time ρ	Finer-unit recovery (%)
TRELLIS.2	Cond.	0.888	0.860	15.42
LumiTex	Distilled	0.978	0.974	1.39
MatLat	Cond.	0.882	0.773	29.74
MatLat	Uncond.	0.931	0.877	32.05

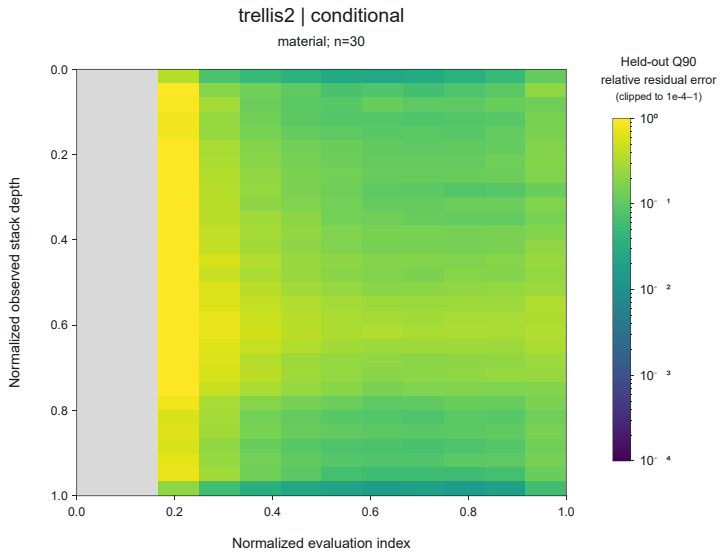

Figure 12: **Whole-block reliability map on TRELLIS.2.** Held-out Q90 residual-prediction error for the conditional branch. Gray cells denote unavailable locations.

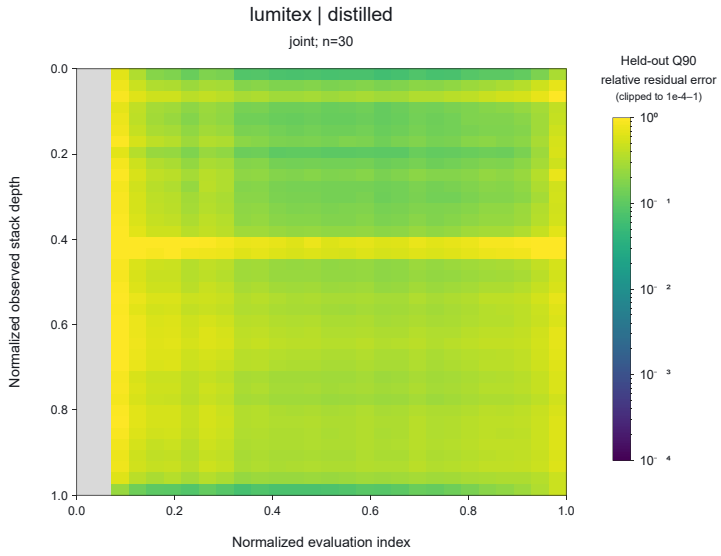

Figure 13: **Whole-block reliability map on LumiTex.** Held-out Q90 residual-prediction error for the distilled joint latent stream. Gray cells denote unavailable locations.

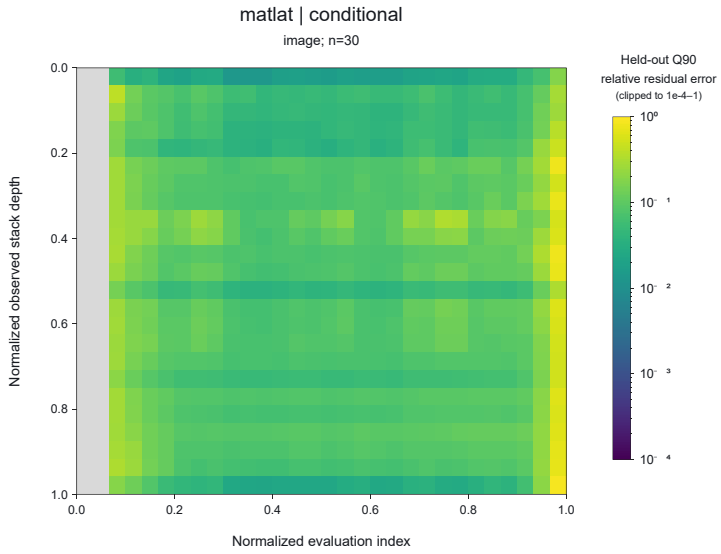

Figure 14: **Whole-block reliability map on MatLat (conditional).** Held-out Q90 residual-prediction error for the conditional branch. Gray cells denote unavailable locations.

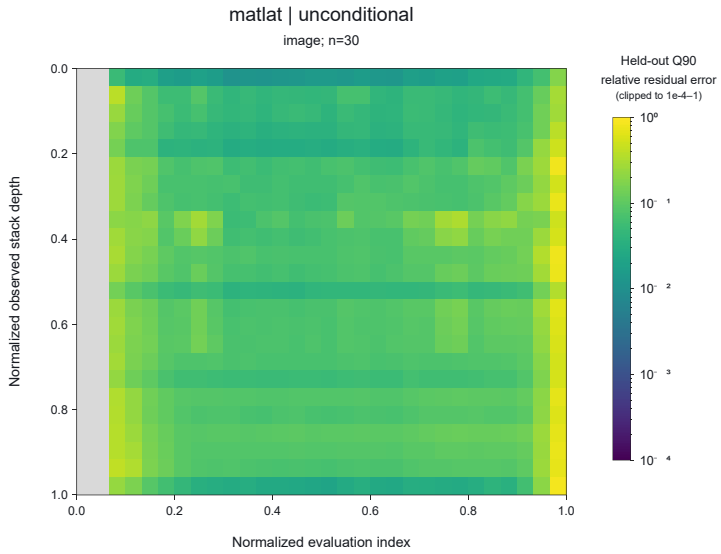

Figure 15: **Whole-block reliability map on MatLat (unconditional).** Held-out Q90 residual-prediction error for the unconditional branch. Gray cells denote unavailable locations.

For this diagnostic only, we additionally test sensitivity to residual normalization. Self normalization uses each target's own residual norm, whereas whole normalization uses the whole-block residual norm as in the main R3D formulation. Table 12 shows that the absolute recovery percentage depends on this choice. We therefore use these diagnostics to establish the recurrence of non-uniform and cross-input reliability structure, rather than to compare recovery percentages directly across architectures.

Table 12: **Normalization sensitivity of cross-model local reliability.** Percentages at $\epsilon = 0.05$. Each normalization calibrates a separate target selection: select Whole when its Q90 passes; otherwise select the passing finer target with the lowest Q90. Recovery credits this frozen finer choice on whole-failure cells. Coverage counts selected cells. These diagnostic target selections do not estimate runtime savings.

Model	Branch	Self-norm. recovery	Whole-norm. recovery	Whole-norm. coverage
TRELLIS.2	Cond.	15.42	79.36	81.67
LumiTex	Distilled	1.39	4.58	4.66
MatLat	Cond.	29.74	21.79	42.86
MatLat	Uncond.	32.05	21.17	41.82

The recurrence claim is strongest for location-dependent structure and cross-input ordering: pooled whole-block correlations span 0.882–0.978 in these probes. Frozen finer-unit recovery is more model-dependent, including only 1.39% for the reported LumiTex self-normalized diagnostic. Thus the data support re-profiling each architecture, rather than expecting equal benefits from the same finer-unit mechanism everywhere.

C.3 Additional Ablations for R3D

All ablations in this section use the same 105 controlled TRELLIS evaluation objects and paired Native references. Unless otherwise stated, shape stage remains exact and only the target texture stage route is changed. LPIPS and nCD are reported in units of 10^{-4} and 10^{-3}, respectively.

Timing-batch provenance. Several ablations reuse the same serialized Mixed-H2 route at $\epsilon = 0.05$ but were timed in separate measurement batches. The main factorial comparison reports $1.222\times$ for this route, while the calibration/threshold batch reports $1.228\times$. The dedicated EAR history experiment is a separate deterministic shared-GPU rerun of the same scheduled route and reports $1.263\times$ for the exact-anchor policy. These latency ratios are therefore retained as batch-specific measurements rather than pooled or forced to a single value; route identity does not imply identical measured wall-clock latency across batches.

C.3.1 RRP: reliability-aware placement

The main paper compares the calibrated route with depth-permuted controls. Each permutation moves complete prediction schedules to different compatible depth locations while preserving the number of prediction actions, target costs, and legal execution masks. Thus the control changes *where* approximation is applied without intentionally changing the amount of scheduled approximation.

The complete numerical comparison is reported in Table 2.

C.3.2 RRP: calibration-set size

We evaluate calibration subsets of 1, 2, 5, 10, and 20 objects. For subset sizes below 20, seeds 17, 43, and 101 define three fixed permutations of the 20 calibration identities and nested prefixes of each permutation are used. These seeds choose calibration identities; they are not generation-noise seeds. The experiment measures sensitivity to the amount and composition of profiling data, while the 105 evaluation objects remain unchanged.

With one calibration object, the three subsets yield LPIPS from 7.330×10^{-4} to 15.689×10^{-4}; the 20-object route yields 5.300×10^{-4}. This supports using multiple objects to reduce sensitivity to a particular calibration identity. The relationship is not strictly monotone for every subset and metric,

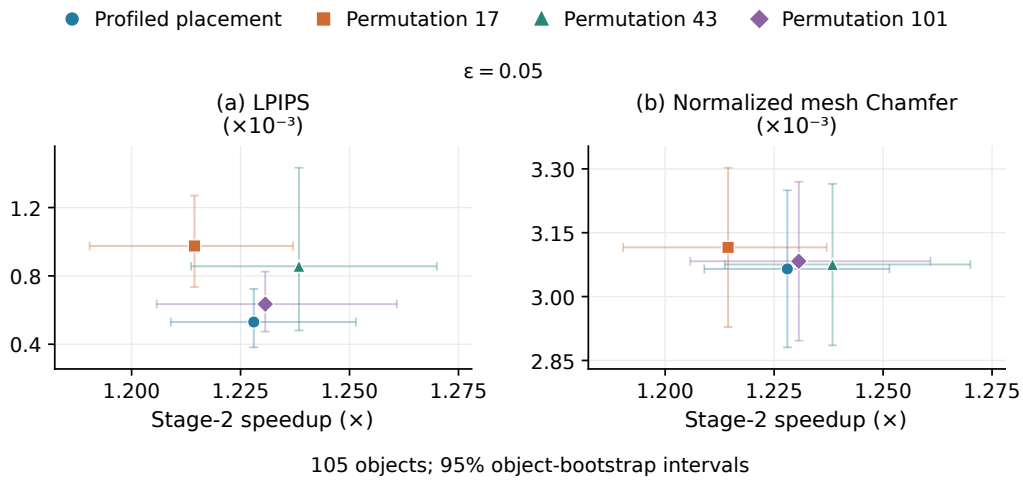

Figure 16: **Reliability-aware placement versus depth permutations.** The calibrated placement produces a better paired-Native quality trade-off at similar target-stage speed.

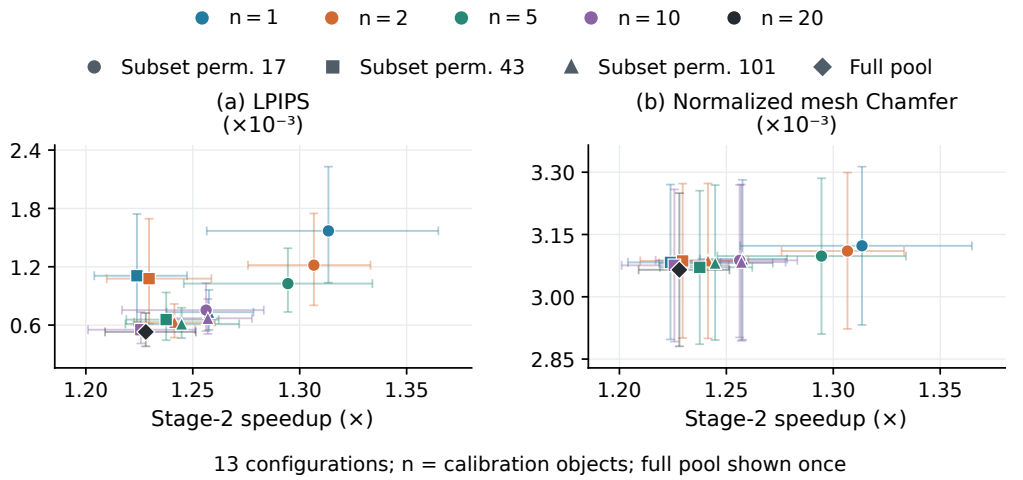

Figure 17: **Sensitivity to calibration-set size at $\epsilon = 0.05$.** Marker shapes denote different calibration subsets. Error bars reflect evaluation-object uncertainty with each compiled route fixed.

and the three identity permutations are not independent repetitions of a new data-collection study. Intervals conditioned on one route do not measure variability from choosing a different calibration set.

C.3.3 GAR: CANDIDATE GRANULARITY AND RELIABILITY THRESHOLD

Whole permits only complete blocks; Grouped permits Prefix/Suffix; Submodule permits SA/CA/MLP; and Mixed permits all six candidate units. Thresholds $\epsilon \in \{0.02, 0.05, 0.10\}$ produce increasingly aggressive operating points. Importantly, equal thresholds do not imply equal runtime because the candidate sets replace different amounts of computation.

At $\epsilon = 0.02$, Whole and Grouped are slightly slower than Native in this batch ($0.985\times$ and $0.988\times$), showing that prediction/history overhead can outweigh sparse skipping opportunities. Mixed reaches $1.082\times$ there and $1.396\times$ at $\epsilon = 0.10$, with the corresponding LPIPS increase reported in the table. This threshold sweep and the joint H1/H2/H3 factorial below are separate measurement batches; small differences at their shared H2 operating point are retained rather than silently averaged.

Table 13: **All calibration-size entries.** Each uses the same 105 evaluation objects. Seed denotes the nested input-ID subset, not generation noise. Normalized Chamfer (nCD) is in 10^{-3}. The 20-object row uses the same serialized Mixed-H2 route as the primary configuration; its speedup is the measurement from this calibration/threshold timing batch.

Cal. objects	Seed	Speedup	LPIPS (10^{-4})	PSNR	nCD ↓
1	17	1.313×	15.689	49.549	3.123
1	43	1.224×	11.074	52.790	3.083
1	101	1.258×	7.330	52.765	3.091
2	17	1.307×	12.165	51.241	3.110
2	43	1.230×	10.777	52.947	3.086
2	101	1.241×	6.297	53.287	3.087
5	17	1.294×	10.268	51.336	3.098
5	43	1.238×	6.565	53.680	3.071
5	101	1.245×	6.120	53.142	3.081
10	17	1.256×	7.539	52.751	3.088
10	43	1.226×	5.525	54.079	3.076
10	101	1.257×	6.700	52.976	3.084
20	–	1.228×	5.300	54.227	3.065

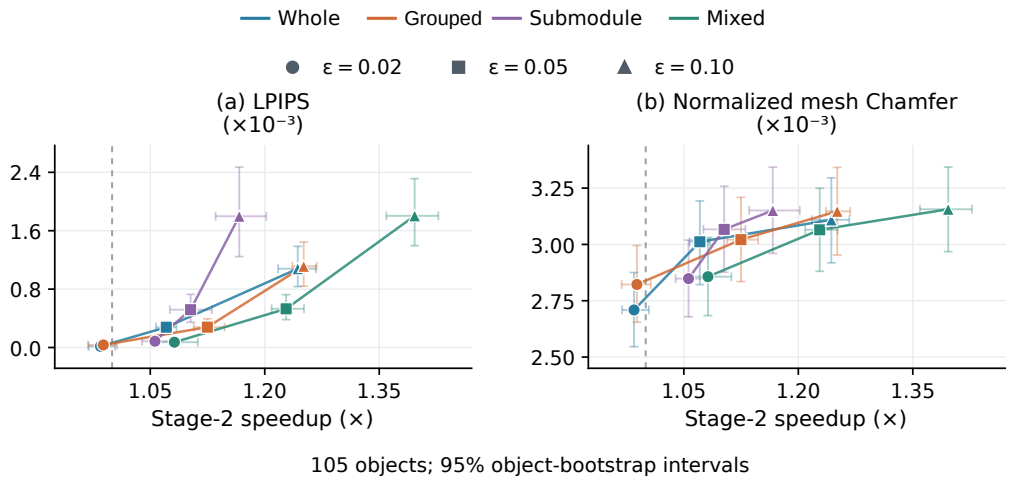

Figure 18: **Granularity and threshold trade-off.** Mixed routing can exploit coarse reliable units where available while retaining finer-grained prediction opportunities elsewhere.

C.3.4 EAR: REFRESH-SCHEDULE FACTORIAL

To isolate the interaction between prediction granularity and exact-anchor refresh, we compile twelve routes from four candidate sets $G \in \{\text{Whole}, \text{Grouped}, \text{Submodule}, \text{Mixed}\}$ and three periodic refresh schedules $R \in \{\text{H1}, \text{H2}, \text{H3}\}$. Each schedule uses reliability measurements collected under the same anchor policy used at inference. All routes use Q90 threshold $\epsilon = 0.05$.

The experiment highlights why EAR is part of route construction rather than a post-hoc runtime knob. Changing H1/H2/H3 changes both the available exact anchors and which candidate residuals pass the reliability criterion. Consequently, a longer nominal prediction horizon need not yield a larger measured speedup: the resulting route can contain more prediction cells while replacing less total component computation.

For a compiled route $\mathcal{M}$, we report the number of prediction cells

$$N_{\text{pred}} = \sum_{c} \mathbf{1}[a_c = \text{Predict}(g_c)] \tag{11}$$

and the replaced-component proxy $\sum_{c:a_c=\text{Predict}(g_c)} C(g_c)$. Fig. 19(b) shows that similar prediction counts can correspond to substantially different replaced computation.

Table 14: **Prediction-granularity candidate sets.** All entries use the same 105 evaluation objects and paired Native reference, with 24 views aggregated per object. Speedup is the ratio of summed per-object median texture stage times. LPIPS is shown in units of 10^{-4}; normalized Chamfer (nCD) is in 10^{-3}. Grouped permits prefix/suffix; Submodule permits SA/CA/MLP; Mixed permits all six targets.

Variant	ϵ	Speedup	LPIPS ↓	PSNR ↑	nCD ↓
Whole	0.02	0.985×	0.163	70.613	2.709
Grouped	0.02	0.988×	0.352	67.294	2.822
Submodule	0.02	1.056×	0.859	64.705	2.848
Mixed	0.02	1.082×	0.725	64.885	2.857
Whole	0.05	1.071×	2.780	57.290	3.012
Grouped	0.05	1.125×	2.789	56.576	3.021
Submodule	0.05	1.103×	5.181	54.771	3.067
Mixed	0.05	1.228×	5.300	54.227	3.065
Whole	0.1	1.243×	10.795	51.209	3.109
Grouped	0.1	1.251×	11.127	48.927	3.147
Submodule	0.1	1.167×	17.982	47.365	3.151
Mixed	0.1	1.396×	18.039	48.065	3.156

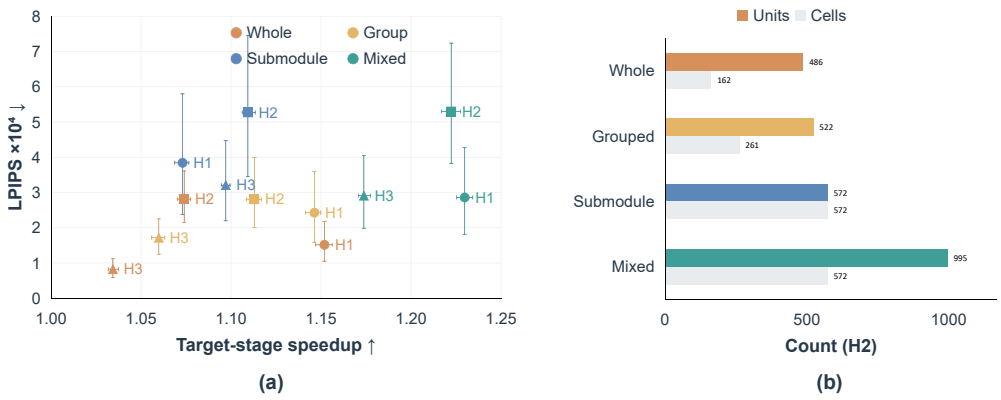

Figure 19: **Granularity and refresh interact through the compiled route.** (a) Four candidate sets with H1/H2/H3 refresh schedules, using matched-history Q90 at $\epsilon = 0.05$; speedups and paired-Native LPIPS use 105 objects with 95% object-bootstrap intervals. Circles/squares/triangles denote H1/H2/H3. (b) H2 counts show that Submodule and Mixed have the same 572 selected cells but replace 572 and 995 component units, respectively. Full counts and metrics appear in Table 15.

Table 15 reports all twelve settings. Mixed-H1 replaces 1,044 component units in 465 cells, while Mixed-H3 replaces 867 units in 581 cells. Their measured speedups are about 1.23× and 1.17×, respectively. Thus a longer nominal horizon and more prediction cells need not produce more replaced component work. Together with Fig. 19, these results show that EAR changes the realizable routing pattern itself rather than only inserting periodic exact evaluations into a fixed route.

C.3.5 EAR: ISOLATING ANCHOR CONTAMINATION AND STALENESS

The schedule factorial changes which candidates can be calibrated and selected. To isolate the history mechanism itself, we additionally fix the complete Mixed-H2 route at $\epsilon = 0.05$ and change only how its anchors are maintained. All three variants retain the same 572 prediction actions, their branch/time/block/target identities, and 995 bypassed component units. Missing-anchor fallback is disallowed. Thus differences cannot be attributed to changing the scheduled network-computation allocation.

Table 15: **Factorial ablation of routing granularity G and refresh schedule R.** Cells denote selected prediction actions and Units denote replaced components. H1/H2/H3 allow one/two/three prediction-eligible evaluations between exact refreshes, respectively. All routes use schedule-matched Q90 with $\epsilon = 0.05$.

G	R	Speedup ($\times$)	LPIPS (10^{-4})	PSNR (dB)	nCD (10^{-3})	Cells	Units
Whole	H1	1.15	1.51	59.98	2.98	258	774
	H2	1.07	2.81	57.28	3.03	162	486
	H3	1.03	0.82	62.73	2.92	107	321
Grouped	H1	1.15	2.43	56.35	3.03	320	640
	H2	1.11	2.81	56.59	3.02	261	522
	H3	1.06	1.72	58.55	3.00	179	358
Submodule	H1	1.07	3.84	55.02	3.02	465	465
	H2	1.11	5.27	54.75	3.07	572	572
	H3	1.10	3.21	56.45	3.04	580	580
Mixed	H1	1.23	2.86	57.86	3.00	465	1044
	H2	1.22	5.30	54.23	3.08	572	995
	H3	1.17	2.91	57.56	3.02	581	867

History policies. EAR commits only the residuals from scheduled exact seed/refresh evaluations. The *predicted-anchor* control also commits the residual predictions actually injected at prediction slots. The *frozen-exact* control retains the initial pair of exact residual values *and their original model-time stamps*. Its later scheduled exact computations still execute and affect the current sampler state, but are not committed as new anchors. Freezing only values while advancing their timestamps would be a different control.

Matched numerical execution. Native and all three history variants are regenerated and timed in this experiment using the same deterministic backend: deterministic PyTorch algorithms are enabled with errors on unsupported operations, and the cuBLAS workspace configuration is `:4096:8`. The audited and timed outputs are required to agree bit-wise within this rerun. Historical outputs from the earlier nondeterministic backend are retained as diagnostics, not used as the new paired Native reference. Each object uses one warming-up and four timed repetitions per arm with balanced interleaving.

Timing scope. The recorded run permits shared-GPU execution. The $1.263\times$, $1.258\times$, and $1.290\times$ ratios therefore describe exploratory timings under that recorded load; their bootstrap intervals do not account for interference from other jobs. The quality comparison and equality of scheduled actions support the history-mechanism analysis. Small latency differences among the policies do not establish a speed advantage of one history policy. This deterministic rerun is a separate timing batch and is not pooled with the $1.222\times$ factorial measurement, the $1.228\times$ calibration/threshold measurement, or the $1.842\times$ deployment setting.

Table 16: **Matched-route EAR comparison.** All policies use 105 objects, 572 prediction cells, and 995 component units. LPIPS is reported in raw units; nCD is reported in units of 10^{-3}. Thus, for example, nCD 3.074 here corresponds to raw nCD 0.003074 in the main-paper EAR table.

History	Stage 2 (s)	Speedup	LPIPS	PSNR	nCD
EAR (exact anchors)	3.236	$1.263\times$	0.000529	54.22	3.074
Predicted anchors	3.248	$1.258\times$	0.003997	42.75	3.274
Frozen initial anchors	3.169	$1.290\times$	0.216588	16.87	4.944

Results. Allowing predicted anchors raises mean LPIPS from 0.000529 to 0.003997 and lowers PSNR by 11.47 dB relative to EAR. Freezing the initial pair has a larger effect: mean LPIPS reaches 0.216588, with a 37.35 dB PSNR reduction. The paired intervals in Table 17 quantify these changes

Table 17: **Paired effect of changing only history.** Control minus EAR, with pointwise 95% intervals from 10,000 paired object-bootstrap replicates. Positive LPIPS/nCD and negative PSNR favor EAR.

Control	Metric	Mean difference	95% interval
Predicted anchors	LPIPS (raw)	0.003468	[0.002914, 0.004142]
Predicted anchors	PSNR (dB)	-11.470	[-12.057, -10.873]
Predicted anchors	nCD (10^{-3})	0.201	[0.176, 0.228]
Frozen initial anchors	LPIPS (raw)	0.216059	[0.192128, 0.241555]
Frozen initial anchors	PSNR (dB)	-37.354	[-38.269, -36.398]
Frozen initial anchors	nCD (10^{-3})	1.871	[1.689, 2.081]

with objects as the independent units. Fig. 20 additionally shows the complete empirical distributions, so the interpretation does not rely only on an aggregate mean.

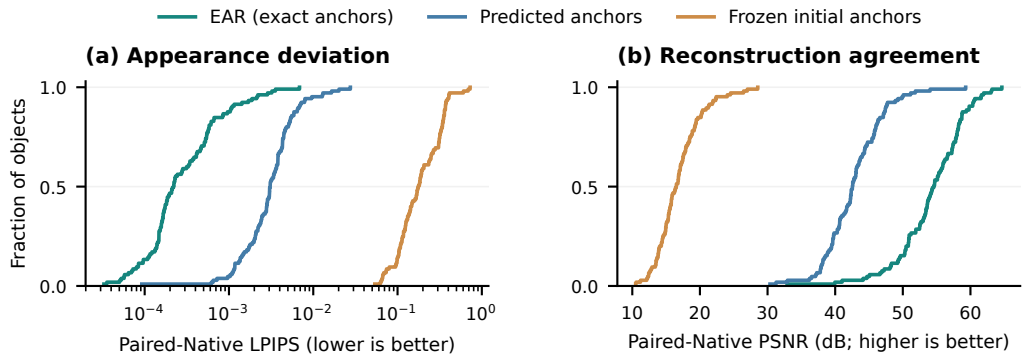

Figure 20: **Object-level error distributions under a fixed route.** Empirical cumulative distributions over the same 105 objects; each LPIPS/PSNR value first averages 24 paired views. The LPIPS axis is logarithmic. These are final-output errors, not unmeasured per-step residual-error trajectories.

The two controls support complementary roles for EAR: avoiding recursively predicted anchors and keeping exact anchors recent. Exact refresh does not erase previously accumulated hidden-state error, and this experiment does not directly measure how residual error propagates at every step. No additional exact mechanism probes were enabled in the recorded run.

C.3.6 Prediction versus direct reuse

R3D predicts residuals by two-anchor linear extrapolation (Eq. (1)). To test whether the gain can be explained by simple caching, we keep the same prediction locations and refresh schedules but replace extrapolation with direct residual reuse. Because the route is calibrated for extrapolation, this is a matched-route mechanism control rather than a fully tuned caching baseline.

At $\epsilon = 0.02$ and 0.05, extrapolation reduces LPIPS relative to reuse on the matched route. At $\epsilon = 0.10$, reuse instead has lower LPIPS (11.936 versus 18.039, both in 10^{-4}) and higher PSNR. This control therefore supports the usefulness of temporal prediction at the moderate operating points, not uniform dominance over direct reuse for every threshold. It also does not replace a separately calibrated and tuned cache baseline.

C.3.7 Why use a frozen route? A causal online-control experiment

We additionally evaluate a causal guard that observes errors only at scheduled exact refreshes and suppresses subsequent candidates when those past errors are large. This guard never queries a skipped location before deciding. It reduces realized predictions from 572 to a mean of 305.05 actions and decreases paired-Native deviation, but its runtime bookkeeping adds approximately 163.68 ms per object and makes this implementation slower than Native. This experiment does

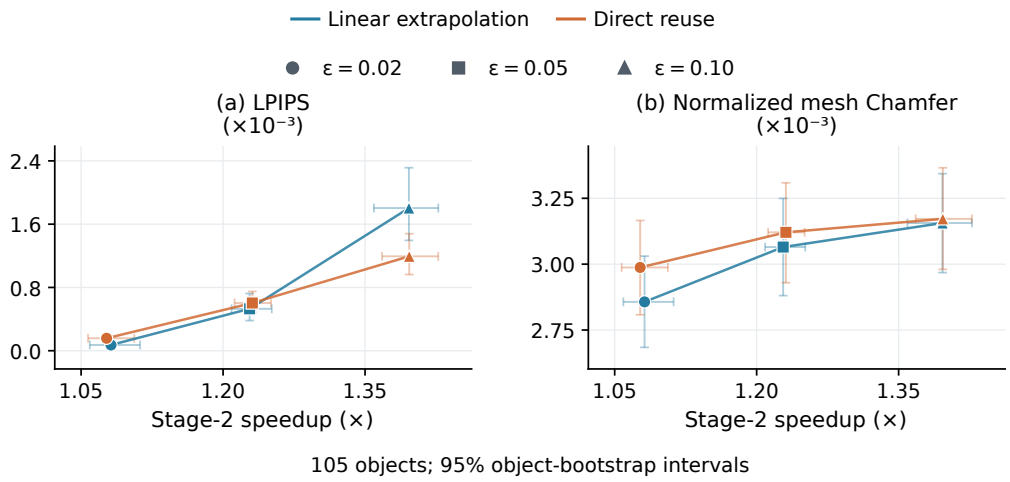

Figure 21: **Linear residual extrapolation versus direct reuse on matched routes.** The route and periodic exact-refresh schedule are fixed; only the runtime residual formula changes.

Table 18: **Prediction versus direct reuse.** All entries use the same 105 evaluation objects and paired Native reference, with 24 views aggregated per object. Speedup is the ratio of summed per-object median texture stage times. LPIPS is shown in units of 10^{-4}; normalized Chamfer (nCD) is in 10^{-3}. Reuse keeps the route calibrated for linear extrapolation and changes only the runtime residual formula; the periodic refresh schedule is unchanged.

Variant	ϵ	Speedup	LPIPS ↓	PSNR ↑	nCD ↓
Linear extrap.	0.02	1.082×	0.725	64.885	2.857
Reuse	0.02	1.077×	1.575	59.212	2.987
Linear extrap.	0.05	1.228×	5.300	54.227	3.065
Reuse	0.05	1.231×	6.035	52.470	3.121
Linear extrap.	0.1	1.396×	18.039	48.065	3.156
Reuse	0.1	1.397×	11.936	48.744	3.172

not rule out better online policies; it explains the practical choice of a low-overhead frozen route in the current implementation.

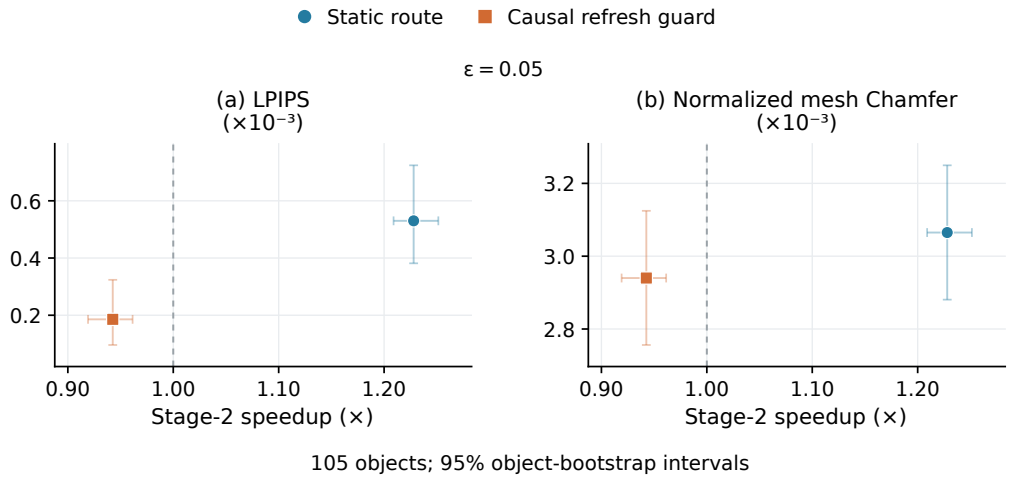

Figure 22: **Frozen routing versus one causal online guard.** Online filtering improves approximation fidelity but loses the acceleration benefit in this implementation because of guard overhead.

Table 19: **Static routing and a causal guard.** All entries use the same 105 evaluation objects and paired Native reference, with 24 views aggregated per object. Speedup is the ratio of summed per-object median texture stage times. LPIPS is shown in units of 10^{-4}; normalized Chamfer (nCD) is in 10^{-3}. The guard filters the same candidate route using past exact-refresh errors; this comparison concerns that specified guard only.

Variant	ϵ	Speedup	LPIPS ↓	PSNR ↑	nCD ↓
Static route	0.05	1.228×	5.300	54.227	3.065
Causal refresh guard	0.05	0.943×	1.857	60.484	2.940

C.3.8 Additional control: low-complexity location models

As a supplementary characterization of Observation 1, we compare whole-block routes constructed from full time–depth profiles with time-only, depth-only, and additive restrictions. Simple profiles can be competitive at some thresholds, while the unrestricted profile provides access to a broader set of location-specific decisions. We therefore interpret the reliability map as containing strong low-complexity structure plus finer cell-specific variation, rather than claiming that arbitrary time–depth interactions are always necessary.

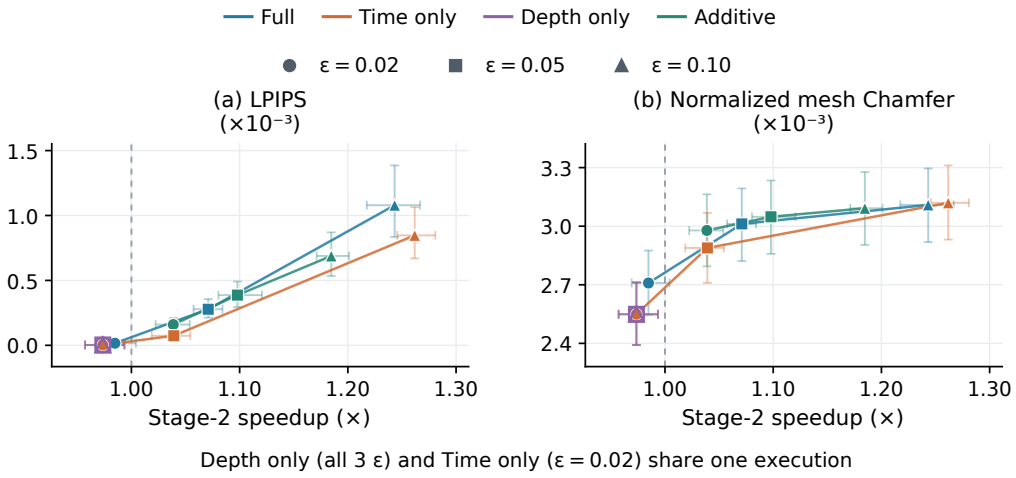

Figure 23: **Whole-block routing under alternative reliability-profile structures.** Curves connect tested thresholds only.

Table 20: **Structure of the routing pattern.** All entries use the same 105 evaluation objects and paired Native reference, with 24 views aggregated per object. Speedup is the ratio of summed per-object median texture stage times. LPIPS is shown in units of 10^{-4}; normalized Chamfer (nCD) is in 10^{-3}. These runs use whole-block targets.

Variant	ϵ	Speedup	LPIPS ↓	PSNR ↑	nCD ↓
Full	0.02	0.985×	0.163	70.613	2.709
Time only	0.02	0.974×	0.030	77.824	2.549
Depth only	0.02	0.974×	0.030	77.824	2.549
Additive	0.02	1.039×	1.599	60.057	2.978
Full	0.05	1.071×	2.780	57.290	3.012
Time only	0.05	1.039×	0.737	63.506	2.889
Depth only	0.05	0.974×	0.030	77.824	2.549
Additive	0.05	1.098×	3.867	56.004	3.048
Full	0.1	1.243×	10.795	51.209	3.109
Time only	0.1	1.262×	8.472	50.761	3.120
Depth only	0.1	0.974×	0.030	77.824	2.549
Additive	0.1	1.185×	6.885	52.669	3.092

C.4 Complete Cross-Backbone Deployment Results

Configuration differences. The TRELLIS deployment row uses a deployment-specific whole-block route with H2 refresh and exact negative-branch CA caching; it is distinct from the Mixed-H2, $\epsilon = 0.05$ route used for controlled mechanism studies. TRELLIS.2 targets material generation and combines residual prediction with exact K/V caching. LumiTex uses a deployment-specific AL-latent route with H1 refresh, while MatLat uses two-anchor extrapolation with damping gain 0.5. These adaptations retain pretrained weights but do not constitute one numerically identical route transferred between models. In particular, exact caching and predictor implementation optimizations contribute to the deployed runtime and are not separately isolated by the headline cross-model speedups.

Table 21 reports absolute Native and R3D wall times for the target stage of each deployment backbone. Table 22 gives the corresponding task-level appearance, geometry, and semantic metrics. These values underlie the compact comparison in Table 1; the interpretation caveat in Appendix B applies to small metric differences.

Table 21: **Target-stage deployment timing on 210 inputs per model.** Times are means of per-input median wall times; speedup is the ratio of their sums. Each model uses its own deployment configuration. TRELLIS uses a whole-block H2 route with exact negative-branch CA caching, distinct from the 105-object mixed-granularity study; TRELLIS.2 includes exact K/V caching; LumiTex uses an AL-latent H1 route; and MatLat uses damped extrapolation. Scopes exclude upstream generation and final asset export.

Model	Target stage	Native (s)	Ours (s)	Speedup
TRELLIS	Stage 2	2.814	1.528	$1.842\times^{\dagger}$
TRELLIS.2	Material	1.834	0.496	$3.698\times^{*}$
LumiTex	AL latent	35.295	28.168	$1.253\times$
MatLat	Material latent	7.020	3.399	$2.066\times$

Table 22: **Task quality on 210 inputs per available configuration.** PSNR and LPIPS compare generated and reference renders; CD is the aligned bidirectional mean unsquared Euclidean nearest-neighbor distance. Uni3D, ULIP-2, and CLIP-I measure semantic consistency with the input image. Each reported metric has 210 valid evaluations. LumiTex and MatLat use shared TRELLIS.2-generated geometry and matched GPU rendering for Native and R3D. These task-level comparisons are distinct from the controlled TRELLIS ablations against paired Native outputs.

Model	Setting	PSNR ↑	LPIPS ↓	CD ↓	Uni3D ↑	ULIP2 ↑	CLIP ↑
TRELLIS	Native	16.293	0.19492	0.06196	0.36540	0.39564	0.86976
TRELLIS	TeaCache	15.907	0.19969	0.06084	0.36765	0.39859	0.87051
TRELLIS	SmoothCache	15.958	0.20000	0.06072	0.36708	0.39855	0.87020
TRELLIS	Ours	15.947	0.19951	0.06083	0.36730	0.39795	0.86999
TRELLIS.2	Native	18.110	0.20586	0.06021	0.34425	0.36711	0.84183
TRELLIS.2	TeaCache	18.138	0.20622	0.06019	0.34394	0.36916	0.84188
TRELLIS.2	SmoothCache	18.179	0.20560	0.06024	0.34414	0.36789	0.83962
TRELLIS.2	Ours	17.931	0.20601	0.06016	0.34062	0.36855	0.83834
MatLat	Native	17.077	0.16697	0.02360	0.30949	0.33255	0.81639
MatLat	Ours	17.104	0.16673	0.02360	0.31023	0.33315	0.81700
LumiTex	Native	12.718	0.34575	0.05037	0.11170	0.11630	0.71180
LumiTex	Ours	12.729	0.34565	0.05037	0.11167	0.11639	0.71186

Reading the quantitative results. For TRELLIS, the Native-to-R3D task changes are -0.346 dB PSNR and $+0.00459$ LPIPS; for TRELLIS.2 they are -0.179 dB and $+0.00015$. The much larger deployed speedup on TRELLIS.2 is a target-stage result under its own material-flow configuration. LumiTex and MatLat show very small changes in the reported appearance/semantic metrics; they use supplied geometry, so unchanged CD in these material-generation comparisons is not evidence of improved geometry generation. No pooled average across backbones is used as a common end-to-end speedup.

All populated task-quality entries contain 210 successful evaluations. The reported values are point estimates under model-specific deployment configurations; small differences between rounded metrics do not establish statistical equivalence.

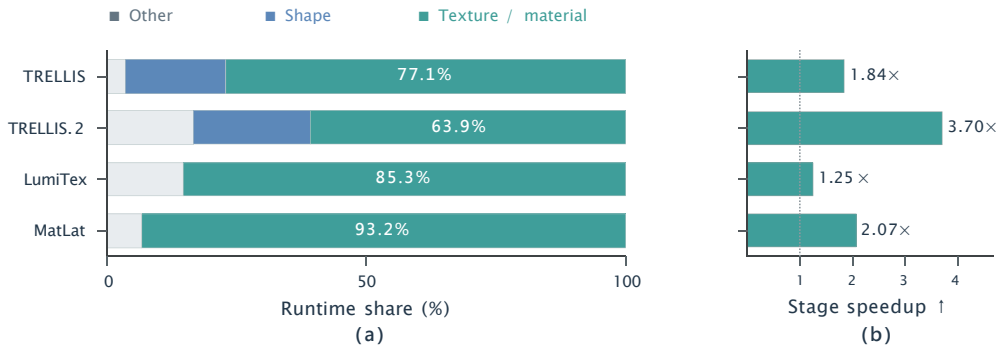

Figure 24: **Pipeline scope and deployment speedups.** (a) Native runtime composition from a separate 105-input profiling run through decoding. (b) Target-stage speedups from the 210-input deployment benchmark. The two panels use separate measurement sets; panel (b) is not an end-to-end speedup derived from panel (a).

C.5 Paired Qualitative Examples

The controlled TRELLIS panel uses the Mixed-H2 route at $\epsilon = 0.05$ and paired Native outputs generated from the same recorded fixtures. We rank objects by mean paired-Native LPIPS over 24 views and visualize a median case, an approximately 90th-percentile case, and the largest-deviation case. The displayed objects are the submarine at rank 53/105, the pig at rank 95/105, and the battery at rank 105/105. The same two camera views (indices 0 and 6) are shown for each Native/Mixed-H2 pair. This selection is intended to expose both typical and tail behavior rather than only favorable examples.



Figure 25: **Paired Native/Mixed-H2 R3D outputs for typical and tail cases.** The accelerated and Native runs share spatial support, initial noise, and conditioning. Objects are selected by their aggregate 24-view paired-Native LPIPS ranking; the displayed cameras are fixed across pairs.

Archived cross-backbone illustrations. The main-paper quality grid uses a TRELLIS gate, the TRELLIS.2 armor example, and a MatLat/LumiTex lantern, with shared display views and detail windows within each output row. The material panel above provides additional detail windows for the same lantern example. These archived examples illustrate appearance and are separate from the LPIPS-ranked 105-object tail selection. Input photographs and generated displays can have different viewpoints; only matched output views should be used for visual comparison among methods. The images are not a substitute for the paired quantitative protocol.



Figure 26: **Material generation with R3D.** MatLat and LumiTex use the same TRELLIS.2-generated input mesh for Native and R3D, with matched output views and detail windows. The comparisons illustrate appearance on fixed geometry.

C.6 Scope of the evidence

Reusable ordering, not universal error bounds. Pooled, fixed-time, and detrended agreement supports shared error ordering over the examined inputs. It does not guarantee equal absolute errors or transfer of an identical route across architectures. Q90 is a calibration statistic; final rollout quality must be measured separately because approximation changes subsequent hidden states.

Compute allocation, not prediction count alone. GAR explains how equally many prediction cells can bypass different component counts. Attention, MLPs, sparse tokens, prediction, and memory traffic have different costs, so the route inventory does not replace measured latency.

Model-specific deployment, not identical interventions. The deployment backbones use different stages and independently configured accelerators, including exact caching and damped extrapolation. Their speedups characterize those implementations. The controlled TRELLIS studies provide the narrower component-level attribution.