CODACACHE: COORDINATED ADAPTIVE BRANCH CACHING FOR EFFICIENT IMAGE-TO-3D GENERATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Image-conditioned diffusion and flow models generate detailed 3D assets from a single image. Full classifier-free guidance (CFG) computes conditional and unconditional predictions at every denoising step, increasing generation cost. The effect of reusing one prediction depends on whether the other is recomputed or reused. In our Hunyuan3D 2.1 diagnostic, reusing both predictions yields a smaller deviation from Full CFG than reusing the conditional prediction while refreshing the unconditional one. At a fixed state and timestep, reuse errors with similar directions and magnitudes can partly offset each other in the CFG prediction. This paper proposes CodaCache, a training-free causal controller that adaptively coordinates CFG branches. It estimates the effects of omitting the unconditional prediction and reusing the conditional one from completed denoising history, using latent motion and time separation. It combines these estimates to choose Full CFG, conditional-only refresh, or conditional-only reuse before current branch evaluation, without extra denoiser calls. On TRELLIS.2 and Hunyuan3D 2.1 with Toys4K and an Objaverse-XL subset, CodaCache uses 37.6–44.7% of Full-CFG denoising FLOPs and achieves 2.34–$2.73\times$ end-to-end generation speedup, excluding offline calibration. Across valid outputs, mean Chamfer distance increases by at most 0.15% and mean F-score decreases by at most 0.10 percentage points relative to Full CFG.

1 INTRODUCTION

Single-image 3D generation has been used in applications such as games and virtual environments (Zhao et al., 2025; Hunyuan3D et al., 2025). In representative systems such as Hunyuan3D 2.1 and TRELLIS.2, encoded image features condition repeated updates of a noisy latent, which is then decoded into geometry (Hunyuan3D et al., 2025; Xiang et al., 2026). To guide each update, a classifier-free guidance (CFG) component is employed to combine two predictions from the same network and latent state, with and without image conditioning (Ho & Salimans, 2022). However, evaluating both prediction branches increases the cost of generation. As a result, accelerating generation while preserving geometric quality has become imperative.

Notably, image and video generation use similar CFG designs. Existing methods mitigate the efficiency bottleneck without retraining the generator. For example, FasterCache evaluates the conditional branch and reconstructs unconditional predictions using cached branch differences (Lyu et al., 2025). For image-to-3D generation, the recently published Fast3Dcache combines token caching with a configured CFG cutoff that disables the unconditional branch (Yang et al., 2026). However, the effect of reusing one prediction on a CFG update depends on whether the other prediction is recomputed or reused. The methods above do not explicitly assess the combined effect of omitting the unconditional prediction and reusing the conditional one when selecting a partial-CFG action.

We now provide experimental evidence of this branch interaction. In Hunyuan3D 2.1, refreshing the unconditional branch does not necessarily improve fidelity to Full CFG when the conditional prediction is reused. Under the same conditional-prediction refresh rule, we compare reusing both conditional (C) and unconditional (U) predictions with pairing cached C with fresh U. Reusing both lowers endpoint Chamfer distance (CD) to Full CFG by at least 30% for each tested object (Figure 3(a)). To examine this interaction, we separately compare reused and fresh predictions at the same latent state and timestep. The C and U reuse errors enter CFG with coefficients of opposite

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

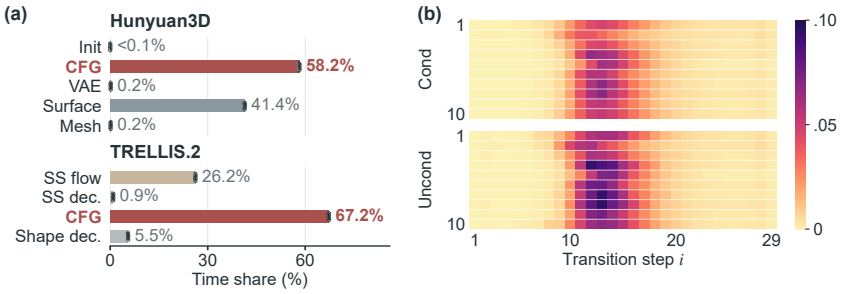

Figure 1: **Denoising cost and temporal reuse potential.** (a) CFG dominates diagnostic core-pipeline runtime in Hunyuan3D and TRELLIS.2; configurations differ from the main comparison (Appendix A). (b) Both branches in Hunyuan3D exhibit small adjacent-step directional changes late in denoising. Both heatmaps use the same color scale for $1 - \cos(v_i, v_{i-1})$.

signs. When their directions and magnitudes are similar, their weighted contributions can partly offset each other, consistent with smaller local errors under joint reuse (Figure 3(b)). Endpoint CD also varies with the number of Full-CFG steps before reuse begins (Figure 2(b)).

We therefore assess U omission and C reuse jointly at each image-to-3D denoising step. Omitting U removes the guidance correction, whereas reusing C introduces an additional prediction error. Figure 1 shows the opportunity for reducing computation: CFG dominates diagnostic core-pipeline runtime, while Hunyuan3D predictions change little in direction between adjacent steps late in denoising.

Motivated by this branch interaction, we introduce CodaCache, a training-free controller that dynamically coordinates CFG branches for image-to-3D generation. Its decisions are causal: it scores Full CFG (F), conditional-only refresh (P), and conditional-only reuse (R) from completed history before evaluating the current branches. Full-CFG steps measure U-omission effects, while C refreshes measure C-reuse error. On a common solver-increment scale, CodaCache estimates these effects from accumulated latent motion and time separation, then adds them when omission and reuse occur together. Offline calibration uses Full-CFG reference runs to fix the distance ranges within which estimates with sufficient history are applied. Among actions meeting these conditions and the score threshold, CodaCache selects the one requiring the fewest branch evaluations; otherwise, it uses Full CFG. Its actions adapt to each input without extra denoiser evaluations or changes to pretrained weights, the native solver, or its update count.

We evaluate CodaCache on two image-to-3D models, Hunyuan3D 2.1 and TRELLIS.2, using Toys4K and Objaverse-XL. The experiments assess geometric quality, computational cost, and the effects of controller components. We also test whether distance ranges calibrated on one dataset transfer to the other. CodaCache uses 37.6–44.7% of Full-CFG FLOPs in the evaluated CFG stages and achieves 2.34–$2.73\times$ end-to-end generation speedup, excluding offline calibration. Mean CD and F-score over valid outputs remain close to those of Full CFG.

2 Related Work

Recent image-to-3D systems generate geometry or textured assets with image-conditioned diffusion or flow models built on compact 3D representations. Direct3D, CLAY, and Hunyuan3D 2.0 model latent geometry (Wu et al., 2024; Zhang et al., 2024; Zhao et al., 2025), while TRELLIS and TRELLIS.2 use structured latents for geometry and appearance (Xiang et al., 2025; 2026). Direct3D-S2 improves scalability through sparse volumetric latents and spatial sparse attention (Wu et al., 2026), and Hunyuan3D 2.1 separates shape generation from material synthesis (Hunyuan3D et al., 2025). CodaCache targets the iterative latent-prediction stage without changing the pretrained generator or its representation.

CFG combines C and U predictions (Ho & Salimans, 2022). Guidance-scheduling methods reduce the cost of these evaluations by restricting or adapting when CFG is applied (Castillo et al., 2025; Kynkäänniemi et al., 2024; Zhang & Wan, 2025; Fu & Li, 2025). Temporal reuse spans feature

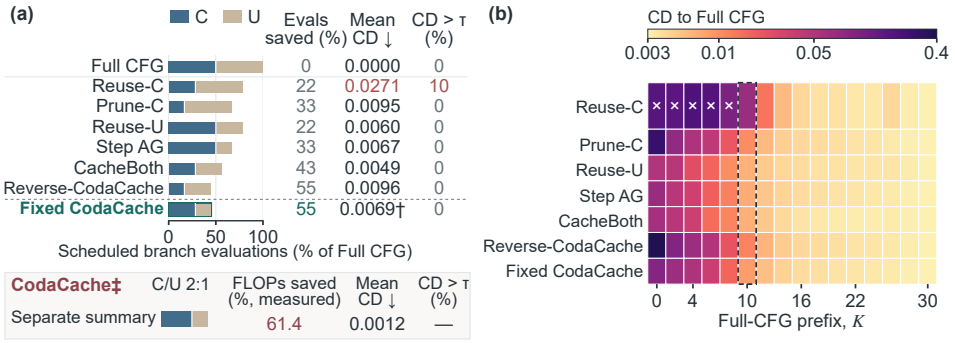

Figure 2: Branch treatment and cutoff sensitivity. (a) In Hunyuan3D, Reuse-C deviates more from Full CFG than Reuse-U at matched scheduled branch savings. Fixed-treatment bars and savings count scheduled C/U evaluations; Fixed CodaCache (†) is an aggregate summary; CodaCache (‡) separately reports measured FLOPs and C/U shares. (b) Seven branch treatments show generally smaller deviations with longer Full-CFG prefixes. Colors encode mean finite CD on a logarithmic scale. The CD threshold is $\tau_{\mathrm{CD}} = 0.05$; $\times$ marks a recorded failed generation excluded from that mean, and the dashed column marks $K = 10$. The residual CD at $K = 30$ is not attributed to the cutoff. All savings are relative to Full CFG.

caching, token reuse, runtime adaptation, and forecasting (Ma et al., 2024; Selvaraju et al., 2024; Zou et al., 2025; Zhang et al., 2025; Liu et al., 2025a; Zhou et al., 2025; Liu et al., 2025b; Feng et al., 2026a). BudCache adapts reuse under budget constraints (Lei et al., 2026), while ToCa studies coordinated token computation across CFG branches (Zou et al., 2025). FasterCache reconstructs U from current C outputs and cached branch differences (Lyu et al., 2025).

BAG learns a budget- and trajectory-conditioned compute/reuse gate whose mask is shared across CFG branches (Zhao et al., 2026). Tortoise and Hare Guidance (THG) estimates a coarse grid for additional guidance while retaining fine-grid C evaluations (Lee et al., 2025). RSTR searches sparse guidance positions and scales, then allocates regional ranks to calibrate feature caching under that schedule (Sun et al., 2026).

Image-to-3D acceleration targets different stages of the generation pipeline. Hash3D reuses diffusion features during score-distillation optimization (Yang et al., 2025), GeoCache transports predicted clean-signal increments across views in texture diffusion (Li et al., 2026), and training-based methods distill multiview or latent-shape generation (Liu et al., 2025c; Lai et al., 2025). E-DiT and Fast-SAM3D reduce computation through routing, caching, or token reduction (Wang et al., 2026; Feng et al., 2026b). Fast3Dcache sets token budgets from voxel-change calibration and token dynamics, with a configured CFG-off cutoff and periodic token corrections (Yang et al., 2026). CodaCache instead selects a complete $F/P/R$ branch action before each native image-to-3D update. Its decision uses two signals available from completed steps: U-omission audits from F actions and C-reuse audits from C refreshes. Their trajectory-based estimates are combined to score candidate actions without an extra denoiser evaluation.

3 Motivation: Reuse Depends on the Other CFG Branch

We study C/U branch interactions in Hunyuan3D using inputs for which the Full-CFG reference run successfully generates a mesh. After evaluating both branches during the initial denoising steps, we compare strategies that recompute, reuse, or omit their predictions.

3.1 Redundancy Identification and Reuse Impact

We first examine temporal redundancy in the two CFG branches. CFG-based denoising, including denoiser evaluations and C/U combination, dominates the measured core-pipeline time of Hunyuan3D and TRELLIS.2 (Figure 1(a)). In Hunyuan3D, both branches show smaller adjacent-step changes in prediction direction late in denoising (Figure 1(b)). These observations motivate testing

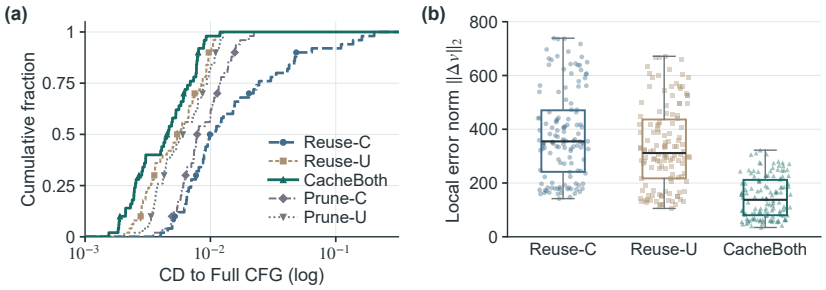

Figure 3: Reuse depends on the other CFG branch. (a) In Hunyuan3D, cumulative endpoint CD distributions relative to Full CFG show a longer high-deviation tail for Reuse-C than for CacheBoth. (b) At matched states and timesteps, CacheBoth has smaller prediction-error norms relative to fresh Full CFG than either single-branch reuse treatment.

prediction reuse to reduce redundant computation, following prior work on diffusion caching (Ma et al., 2024; Liu et al., 2025a).

We next examine how branch reuse affects the final geometry. Directional similarity alone cannot capture magnitude changes or C/U interactions, so we measure endpoint Chamfer distance (CD) to Full CFG. At matched savings in scheduled branch evaluations, Reuse-C produces larger deviations than Reuse-U (Figure 2(a)). Reuse-C combines cached C with fresh U, whereas CacheBoth reuses both branches. Under the same C-refresh rule, CacheBoth has lower endpoint CD than Reuse-C for all compared objects (Figure 3(a)). Thus, fresh U does not necessarily bring the output closer to Full CFG in this Hunyuan3D comparison. In the separate ground-truth evaluation, CacheBoth achieves lower CD and higher F-score than Reuse-C on both backbones (Table 2). These endpoint comparisons do not isolate the cause because trajectories differ. We next examine aligned branch errors to interpret the observed interaction.

3.2 Prediction errors introduced by reuse

This analysis interprets the observed branch interaction. CodaCache neither assumes nor predicts cancellation online. At the same latent state and timestep, the fresh CFG prediction is $v_{\text{cfg}} = sv_c - (s-1)v_u$ (Ho & Salimans, 2022), where $s > 1$ is the guidance scale. Let $e_c = v_c - \widetilde{v}_c$ and $e_u = v_u - \widetilde{v}_u$ denote the errors introduced by reusing C and U, respectively. The resulting prediction errors are

$$\begin{aligned} \Delta v_{\text{Reuse-C}} &= -se_c, \\ \Delta v_{\text{Reuse-U}} &= (s-1)e_u, \\ \Delta v_{\text{CacheBoth}} &= -se_c + (s-1)e_u. \end{aligned} \tag{1}$$

Under joint reuse, the error terms $-se_c$ and $(s-1)e_u$ have coefficients of opposite signs. When e_c and e_u have similar directions and magnitudes, these weighted terms can partly offset each other. In the limiting case $e_c = e_u$, CacheBoth yields $\Delta v = -e_c$, rather than the $-se_c$ of Reuse-C.

In a separate diagnostic at matched states and timesteps, CacheBoth has smaller prediction-error norms than either single-branch reuse treatment (Figure 3(b)). The branch errors have a mean cosine similarity of 0.9259, consistent with partial cancellation. Because later states may diverge, the local analysis does not predict or bound endpoint deviation.

3.3 Effect of Full-CFG prefix length

In this Hunyuan3D diagnostic, short Full-CFG prefixes produce the largest deviations, while differences between treatments generally narrow as prefix length K increases (Figure 2(b)). Increasing K delays approximate updates and reduces their number. The sweep cannot separate these effects or establish a universal cutoff.

These diagnostics show that fidelity to Full CFG depends on branch treatment and approximation timing. This motivates scoring complete actions using trajectory history.

4 CODACACHE

At each native solver update, CodaCache uses completed trajectory history to score three nested actions: Full CFG (F), C-only refresh (P), and C-only reuse (R) (Figure 4). An approximate action is eligible only with sufficient valid history and trajectory distance within its calibrated support interval. The resulting sequence adapts to each sample while retaining the pretrained denoiser, native solver, and all solver updates.

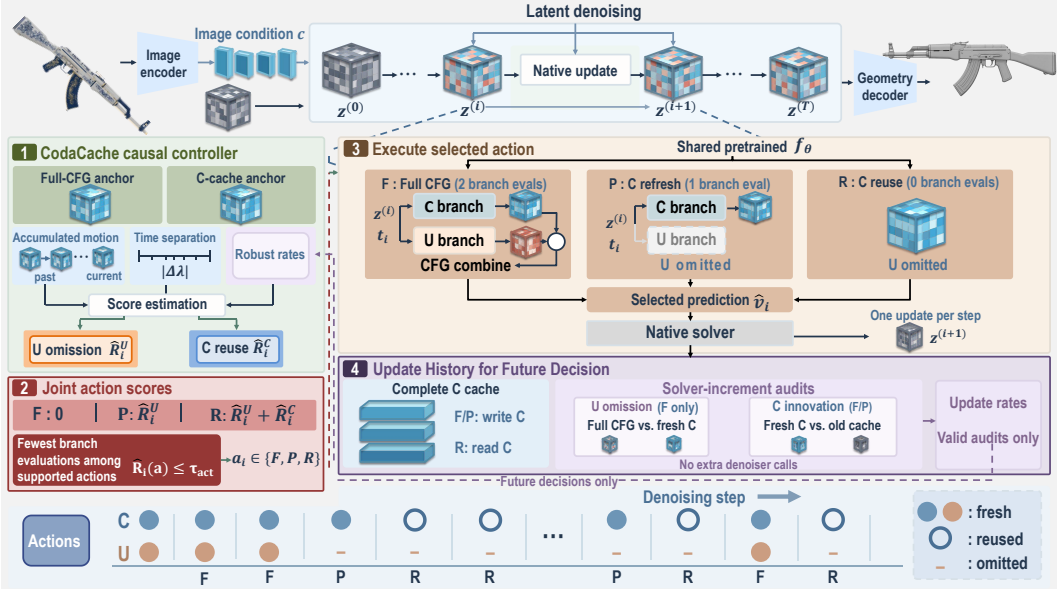

Figure 4: Overview of CodaCache for image-to-3D generation. (1) The causal controller estimates U-omission and C-reuse scores from completed trajectory history. (2) Using joint action scores, it selects the eligible $F/P/R$ action with the fewest branch evaluations under the fixed score threshold. (3) Executing the selected action provides a prediction for one native solver update. (4) Fresh C predictions update the cache, and valid audits update rate estimates used only in subsequent decisions, without extra denoiser evaluations. The trace begins with three F actions, meeting the $n_{\min} = 2$ history-count condition when their rates are valid.

4.1 JOINT ACTIONS AND CONTROLLER OVERVIEW

Given image condition $c = E(I)$ and unconditional input $c_\emptyset$, let $z^{(i)}$ denote the latent state before update $i \in \{0, \dots, T-1\}$. The pretrained denoiser produces

$$v_{c,i} = f_\theta(z^{(i)}, t_i, c), \qquad v_{u,i} = f_\theta(z^{(i)}, t_i, c_\emptyset), \qquad v_{F,i} = s v_{c,i} - (s-1) v_{u,i}, \tag{2}$$

where s is the guidance scale and $v_{F,i}$ is the fresh Full-CFG prediction.

We express each action through its native solver increment:

$$\Delta_i(v; h_i) := \mathcal{S}_i(z^{(i)}, h_i, v) - z^{(i)}, \qquad z^{(i+1)} = z^{(i)} + \Delta_i(\widehat{v}_i; h_i). \tag{3}$$

Here h_i contains any multistep history. All candidate actions use the same $(z^{(i)}, t_i, h_i)$ and differ only in their prediction $\widehat{v}_i$.

At each update, F evaluates fresh C and U; P evaluates fresh C and omits U; and R reuses a complete cached C while omitting U. Their solver predictions and branch-equivalent costs are

$$\big(\widehat{v}_i, \text{cost}(a_i)\big) = \begin{cases} (v_{F,i}, 2), & a_i = F, \\ (v_{c,i}, 1), & a_i = P, \\ (\bar{v}_{c,j}, 0), & a_i = R. \end{cases} \tag{4}$$

Whenever C is recomputed, it replaces the cached C prediction. Action R requires a valid cached C from an earlier update $j < i$; internal activations and CFG residuals are not cached. CacheBoth is a diagnostic intervention, not a controller action. Its comparison with Reuse-C motivates coordinated treatment but does not determine which partial CFG action to take. Relative to F, P omits U and R additionally reuses C. We score these changes on a common solver scale without assuming directional cancellation. Section 5.4 evaluates the contribution of each approximation.

At a fixed solver state, the R–F increment difference decomposes exactly through P (Appendix B.2). The triangle inequality on a shared scale motivates adding the U-omission and C-reuse scores, estimated from completed history in Section 4.2:

$$\widehat{R}_i(F) = 0, \qquad \widehat{R}_i(P) = \widehat{R}_i^U, \qquad \widehat{R}_i(R) = \widehat{R}_i^U + \widehat{R}_i^C. \tag{5}$$

Historical extrapolation of these components is empirical. Let $\mathcal{A}_i^{\text{legal}}$ contain F and P, and also R when a valid C cache exists. Before evaluating the current branches, the controller selects the least expensive legal action whose score is at most τ_{act}:

$$a_i = \arg\min_{a \in \mathcal{A}_i^{\text{legal}}} \text{cost}(a) \quad \text{subject to} \quad \widehat{R}_i(a) \le \tau_{\text{act}}. \tag{6}$$

For finite, supported scores and a valid C cache, R passes exactly when $\widehat{R}_i^C \le \tau_{\text{act}} - \widehat{R}_i^U$: a larger U-omission estimate leaves less score margin for C reuse. This per-update margin is neither a cumulative budget nor a bound on accumulated latent or endpoint geometry error. If R fails, P is selected when its U score passes; otherwise F is used. A later F does not undo deviations from earlier approximate actions.

No cutoff K or fixed reuse count r is supplied; F may recur after U omission. Appendix B.5 defines summaries of the resulting trace.

4.2 Causal trajectory scores

For any tensor x, define $\|x\|_{\text{RMS}} = (\|x\|_2^2 / \text{numel}(x))^{1/2}$. Whenever C is evaluated, its solver increment defines a common normalization scale for both measurements:

$$\nu_i = \|\Delta_i(v_{c,i}; h_i)\|_{\text{RMS}} + \epsilon. \tag{7}$$

Executing F provides both C and U, allowing us to measure the effect of U omission on the solver update:

$$q_i^U = \frac{\|\Delta_i(v_{F,i}; h_i) - \Delta_i(v_{c,i}; h_i)\|_{\text{RMS}}}{\nu_i}. \tag{8}$$

The additional increment requires only solver arithmetic. Thus q_i^U directly measures the normalized effect of replacing Full CFG by fresh C at an executed F anchor and becomes available only after that action.

To use these observations at later updates, we measure how far the trajectory has moved from an earlier anchor $j < i$:

$$D_{i|j} = \sum_{k=j}^{i-1} \frac{\|z^{(k+1)} - z^{(k)}\|_{\text{RMS}}}{\|z^{(k)}\|_{\text{RMS}} + \epsilon} + \eta \frac{|\lambda_i - \lambda_j|}{\lambda_{\max} - \lambda_{\min} + \epsilon}. \tag{9}$$

Here $\lambda_i \in [0, 1]$ is the generator's normalized native time coordinate. Eq. (9) combines latent motion with time separation, both available before update i for causal scoring.

For a Full-CFG action k, let $p_F(k)$ denote the previous valid Full-CFG anchor. The change in the U-omission observation per unit trajectory distance is

$$\ell_k^U = \frac{|q_k^U - q_{p_F(k)}^U|}{D_{k|p_F(k)} + \epsilon}. \tag{10}$$

Valid values from Eq. (10) update the robust rate estimate $L_i^{U,+}$ using an exponential moving average and a scaled recent median absolute deviation. Appendix B.3 gives the update rule, window, and validity checks.

Let $f(i)$ be the latest valid Full-CFG anchor available before update i. The U-omission score is

$$\widehat{R}_i^U = q_{f(i)}^U + L_i^{U,+} D_{i|f(i)}. \tag{11}$$

The first term is the measured omission effect at the latest anchor; the second adds a margin for trajectory drift since that anchor. Insufficient valid history or a distance outside the fixed support interval sets the score to $+\infty$. Since each rate compares two anchors, U omission requires $n_{\min}$ valid rates from at least $n_{\min} + 1$ valid Full-CFG observations.

We estimate C-reuse error in the same way. Every F or P action refreshes C. If the old cache $\bar{v}_{c,p}$ was written at $p < i$, we compare its solver increment with that of the fresh C prediction before overwriting the cache:

$$y_i^C = \frac{\|\Delta_i(v_{c,i}; h_i) - \Delta_i(\bar{v}_{c,p}; h_i)\|_{\mathrm{RMS}}}{\nu_i}, \qquad \ell_i^C = \frac{y_i^C}{D_{i|p} + \epsilon}. \tag{12}$$

This audit also uses solver arithmetic without an extra denoiser evaluation.

Let $L_i^{C,+}$ be the robust rate estimate from valid ℓ_i^C values. For a cache last refreshed at $j < i$, the predicted C-reuse score is

$$\widehat{R}_i^C = L_i^{C,+} D_{i|j}. \tag{13}$$

There is no intercept because C-reuse discrepancy is zero when a fresh cache is written; the score grows with distance from that anchor. An unavailable rate, invalid cache, or unsupported distance sets $\widehat{R}_i^C = +\infty$, excluding R while leaving P feasible if $\widehat{R}_i^U \le \tau_{\mathrm{act}}$.

4.3 Causal execution, fallback, and fixed configuration

At update i, the controller selects a_i from completed history and then executes it. Valid audits update rate estimates only for subsequent decisions. An unavailable U-omission score forces F. When C reuse is unsupported, the controller selects P if $\widehat{R}_i^U \le \tau_{\mathrm{act}}$ and F otherwise. No counterfactual branch evaluation, endpoint geometry, or rollout is used online. Controller constants and per-backbone support intervals are frozen; only estimator states and actions adapt to the sample trajectory. Appendix B.1 gives the complete execution rules.

4.4 Execution and measurement

Every selected action retains one native solver update, preserving the original number of denoising steps. We measure FLOPs over the evaluated CFG denoising stages relative to Full CFG within each model–dataset setting. End-to-end generation time starts after image preprocessing and ends at generation completion, including solver and controller overhead.

5 Experiments

We evaluate CodaCache's geometric quality and computational efficiency, and conduct ablation studies of its key controller components.

5.1 Experimental setup

We evaluate TRELLIS.2 (Xiang et al., 2026) and Hunyuan3D 2.1 (Hunyuan3D et al., 2025) on all 4,179 Toys4K (Stojanov et al., 2021) objects from 105 categories and 3,000 Objaverse-XL (Deitke et al., 2023) objects from 200 categories. CodaCache controls TRELLIS.2's two Shape SLat (structured latent) cascade levels (eight Euler updates each) and Hunyuan3D's Shape DiT (30 Euler updates). For the main comparisons, each backbone's support intervals are calibrated offline from both datasets' pooled Full-CFG trajectories, then fixed during inference. Cross-dataset transfer uses only the other dataset for calibration, with matched budgets and unchanged controller settings. This changes CD by at most 0.0003, and F-score, FLOPs, and valid-output rate by at most 0.32, 1.6, and 0.23 percentage points, respectively (Table 6). We compare with Full CFG and eleven baselines grouped by their main mechanism: guidance scheduling or approximation (Step AG (Zhang & Wan, 2025), Adaptive Guidance (Castillo et al., 2025), Limited-Interval Guidance (Kynkäänniemi et al.,

2024), THG (Lee et al., 2025), TCFG (Fu & Li, 2025), and StreamDiffusion R-CFG (Kodaira et al., 2025)) and caching (RSTR (Sun et al., 2026), Fast3Dcache (Yang et al., 2026), SenCache (Haghighi & Alahi, 2026), ResCa (Fang et al., 2026), and DPCache (Cui et al., 2026)). All experiments use an NVIDIA RTX A6000 with matched inputs, backbone settings, geometry evaluation, and timing within each setting. Ablations change only the specified component or fixed policy. We report five-run means; Appendix C gives detailed configurations, metrics, and timing procedures.

Table 1: Ground-truth geometry quality and CFG-stage FLOPs relative to Full CFG. CD and F-score (%, $\tau_F = 0.05$) are averaged over each method's valid outputs. Bold and underline mark the best and second-best distinct unrounded values.

	Toys4K						Objaverse-XL					
	TRELLIS.2			Hunyuan			TRELLIS.2			Hunyuan		
Method	CD ↓	F-s ↑	FLOPs (%) ↓	CD ↓	F-s ↑	FLOPs (%) ↓	CD ↓	F-s ↑	FLOPs (%) ↓	CD ↓	F-s ↑	FLOPs (%) ↓
Full CFG	**0.0686**	54.82	100.0	**0.0159**	**75.36**	100.0	**0.0261**	**85.86**	100.0	0.0590	64.58	100.0
Step AG	0.0793	50.69	65.7	0.0235	69.28	66.9	0.0295	83.14	61.8	0.0692	64.48	61.4
Adaptive Guidance	0.0761	53.08	52.1	0.0194	70.34	63.0	0.0357	81.27	49.0	0.0628	57.39	64.0
Limited-Interval	0.0773	53.37	50.2	0.0207	69.97	52.5	0.0314	82.67	45.2	0.0814	62.84	51.5
THG	0.0711	51.68	68.0	0.0185	72.49	55.8	0.0301	81.77	51.5	0.0754	59.10	77.7
TCFG	0.0759	53.22	49.5	0.0197	73.22	54.2	0.0300	83.91	47.1	0.0891	61.29	60.4
StreamDiff. R-CFG	0.0741	53.45	64.0	0.0211	73.68	46.7	0.0313	81.34	48.8	0.0674	55.44	47.3
RSTR	0.0699	52.49	47.8	0.0187	74.94	50.1	0.0305	82.69	50.1	0.0732	58.72	67.5
Fast3Dcache	0.0693	54.09	56.3	0.0203	73.47	57.9	0.0282	83.33	46.2	0.0599	64.10	50.6
SenCache	0.0708	53.12	57.2	0.0173	71.58	46.5	0.0275	84.16	61.9	0.0849	60.35	57.1
ResCa	0.0701	53.77	51.2	0.0168	72.61	56.5	0.0284	82.91	48.8	0.0602	63.52	60.1
DPCache	0.0698	53.56	53.1	0.0160	73.19	50.3	0.0270	84.91	45.9	0.0786	56.98	56.5
CodaCache	0.0687	**54.84**	**43.8**	0.0159	75.35	**44.7**	0.0262	85.76	**39.6**	**0.0590**	**64.62**	**37.6**

5.2 Adaptive acceleration and geometric quality

CodaCache uses 37.6–44.7% of Full-CFG denoising FLOPs and achieves a 2.34–2.73× generation speedup, excluding offline calibration (Table 1, Figure 5). Across the four settings, mean CD and F-score over valid outputs remain close to those of Full CFG: unrounded means show at most a 0.15% relative CD increase and a decrease of 0.10 percentage points in F-score. The valid-output rates of Full CFG and CodaCache differ by at most 0.40 percentage points (Appendix C).

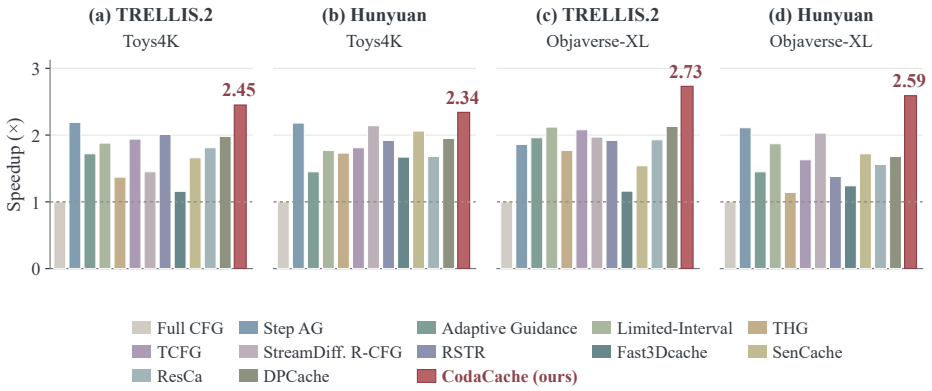

Figure 5: Speedup over matched all-step Full CFG (1×, dashed line). Generation time is measured from the completion of image preprocessing to generation completion, excluding offline calibration.

5.3 Fixed-policy evaluation

After the Full-CFG prefix, Reuse-C/U reuses the named prediction and refreshes the other, while CacheBoth reuses both (Table 2). Specifically, Fixed CodaCache-r omits U and follows fresh C with up to r reuse updates; Reverse-CodaCache omits C and periodically reuses U. Prune-C retains fresh U; Last-/EMA-Delta reuse the latest or smoothed guidance residual.

CacheBoth and Reuse-U improve both geometry metrics over Reuse-C; CacheBoth also reduces FLOPs. Reverse-CodaCache improves both metrics over Fixed CodaCache-r2 except for a 0.49-percentage-point F-score decrease on TRELLIS.2/Toys4K. On Hunyuan, $r = 5$ and 6 reduce FLOPs but raise CD and lower F-score relative to $r = 4$. CodaCache improves both metrics over fixed $r = 2$, with 9.0–17.4% lower FLOPs except on Hunyuan3D/Toys4K (5.9% higher; Table 1).

Table 2: Fixed-policy comparisons of branch treatment and reuse count. FLOPs are normalized to Full CFG. Bold and underline mark the best and second-best distinct unrounded values.

	Toys4K						Objaverse-XL					
	TRELLIS.2			Hunyuan			TRELLIS.2			Hunyuan		
Method	CD ↓	F-s ↑	FLOPs (%) ↓	CD ↓	F-s ↑	FLOPs (%) ↓	CD ↓	F-s ↑	FLOPs (%) ↓	CD ↓	F-s ↑	FLOPs (%) ↓
Full CFG	**0.0686**	**54.82**	100.0	0.0159	**75.36**	100.0	**0.0261**	**85.86**	100.0	0.0590	**64.58**	100.0
Reuse-C	0.0820	51.05	82.2	0.0169	72.22	77.3	0.0312	82.09	76.3	0.0627	61.44	79.3
Reuse-U	0.0690	54.70	75.2	**0.0159**	73.40	76.3	0.0263	84.74	76.6	**0.0589**	62.62	75.4
CacheBoth	**0.0686**	52.92	63.5	0.0159	74.36	55.7	**0.0261**	84.79	52.5	0.0590	63.58	54.7
Fixed CodaCache-r1	0.0739	50.71	59.2	0.0177	72.27	48.7	0.0303	82.45	51.0	0.0641	62.55	53.0
Fixed CodaCache-r2	0.0724	54.11	53.0	0.0170	73.26	42.2	0.0312	81.76	43.5	0.0617	60.62	42.7
Fixed CodaCache-r4	0.0768	48.22	**42.8**	0.0180	71.39	41.1	0.0349	78.26	**42.9**	0.0597	59.75	39.9
Fixed CodaCache-r5	–	–	–	0.0265	63.19	32.2	–	–	–	0.0812	52.91	31.4
Fixed CodaCache-r6	–	–	–	0.0351	57.64	**29.7**	–	–	–	0.0937	47.56	**28.1**
Prune-C	**0.0686**	54.12	58.8	0.0160	73.66	56.7	0.0283	83.86	58.2	0.0597	61.58	56.7
Reverse-CodaCache	**0.0686**	53.62	48.8	0.0160	74.36	46.4	0.0277	84.86	47.5	0.0596	62.58	44.3
Last-Delta	0.0686	52.94	61.2	0.0159	74.53	58.3	0.0282	83.98	67.3	0.0595	63.40	58.3
EMA-Delta	0.0687	52.29	61.2	0.0159	73.36	68.3	0.0271	84.74	57.3	0.0594	62.58	58.3

5.4 Adaptive-controller component ablations

Separate acceptance is an internal ablation that applies the threshold to each component score instead of their sum. It and the variants removing P, R, or either distance term use more FLOPs, with higher CD and lower F-score, than CodaCache (Table 3).

In the Hunyuan3D 2.1/Toys4K diagnostic, we compute paired differences in local error (P minus R) at saved states where the rules disagree. Their medians are -0.0141 and -0.0309 on CodaCache and Separate acceptance trajectories, respectively. Candidate-R errors exceed 0.15 in $21/3652$ and $275/3556$ steps (Appendix F, Table 11(a)). These same-state comparisons assess local selection; full-run FLOPs and geometry are measured independently. Appendix E reports sensitivity.

Table 3: Adaptive-controller ablations. Separate acceptance selects R when both scores are at most 0.15 and C reuse is supported. Otherwise, it selects P if the U-omission score meets this bound, and F if not. Removing R or P disables C-only reuse or refresh, respectively; distance variants remove one term. Metrics follow Table 1; see Appendix B.4 for calibration.

	Toys4K						Objaverse-XL					
	TRELLIS.2			Hunyuan			TRELLIS.2			Hunyuan		
Method	CD ↓	F-s ↑	FLOPs (%) ↓	CD ↓	F-s ↑	FLOPs (%) ↓	CD ↓	F-s ↑	FLOPs (%) ↓	CD ↓	F-s ↑	FLOPs (%) ↓
Full CFG	**0.0686**	54.82	100.0	**0.0159**	**75.36**	100.0	**0.0261**	**85.86**	100.0	0.0590	64.58	100.0
Separate acceptance	0.0724	51.94	47.9	0.0207	73.68	50.8	0.0288	81.28	43.5	0.0635	64.10	44.0
w/o R (F/P only)	0.0717	50.49	71.3	0.0197	67.19	71.9	0.0294	73.79	82.6	0.0646	61.97	70.4
w/o P (F/R only)	0.0699	52.67	60.7	0.0175	74.38	52.3	0.0302	84.64	63.1	0.0603	62.49	62.6
w/o latent drift	0.0734	53.11	53.2	0.0168	74.97	49.3	0.0296	83.22	54.8	0.0613	63.48	53.5
w/o time distance	0.0704	54.01	50.8	0.0161	75.12	48.2	0.0278	84.88	45.2	0.0601	62.19	46.3
CodaCache	0.0687	**54.84**	**43.8**	0.0159	75.35	**44.7**	0.0262	85.76	**39.6**	**0.0590**	**64.62**	**37.6**

6 Conclusion

Observed branch interaction motivates CodaCache, a training-free controller for image-to-3D generation. It uses completed denoising history to jointly assess the effects of omitting unconditional predictions and reusing conditional ones. Across four settings, it uses 37.6–44.7% of the denoising floating-point operations required by full classifier-free guidance, with similar mean geometric quality over valid outputs. Calibrating on the other evaluated dataset retains this trade-off.

AI USE STATEMENT

Generative AI tools were used to assist with manuscript editing, including language polishing and improvements to clarity and organization, and to check LaTeX formatting. The authors independently identified and selected the cited literature, reviewed all AI-assisted revisions, and take full responsibility for the final content.

ETHICS STATEMENT

This work uses publicly available datasets and pretrained models and does not involve human-subject studies. We have identified no ethical concerns specific to this study.

REPRODUCIBILITY STATEMENT

Implementation details and experimental settings are provided in Section 4 and Appendices B and C. Code will accompany the submission as supplementary material and will subsequently be released publicly.

REFERENCES

Angela Castillo, Jonas Kohler, Juan C Pérez, Juan Pablo Pérez, Albert Pumarola, Bernard Ghanem, Pablo Arbeláez, and Ali Thabet. Adaptive guidance: Training-free acceleration of conditional diffusion models. *Proceedings of the AAAI Conference on Artificial Intelligence*, 39(2):1962–1970, 2025.

Bowen Cui, Yuanbin Wang, Huajiang Xu, Biaolong Chen, Aixi Zhang, Hao Jiang, Zhengzheng Jin, Xu Liu, and Pipei Huang. Denoising as path planning: Training-free acceleration of diffusion models with dpcache. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 43632–43642, June 2026.

Matt Deitke, Ruoshi Liu, Matthew Wallingford, Huong Ngo, Oscar Michel, Aditya Kusupati, Alan Fan, Christian Laforte, Vikram Voleti, Samir Yitzhak Gadre, et al. Objaverse-xl: A universe of 10m+ 3d objects. *Advances in Neural Information Processing Systems*, 36:35799–35813, 2023.

Haipeng Fang, Yu Li, Fan Tang, Yixing Lu, Juan Cao, and Sheng Tang. Resca: Residual caching for diffusion transformers acceleration. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 32957–32966, 2026.

Liang Feng, Shikang Zheng, Jiacheng Liu, Yuqi Lin, Qinming Zhou, Peiliang Cai, Xinyu Wang, Junjie Chen, Chang Zou, Yue Ma, et al. Hicache: A plug-in scaled-hermite upgrade for taylor-style cache-then-forecast diffusion acceleration. In *International Conference on Learning Representations*, volume 2026, pp. 112361–112386, 2026a.

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

Xiaomeng Fu and Jia Li. Tcfg: Truncated classifier-free guidance for efficient and scalable text-to-image acceleration. In *2025 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 18552–18562. IEEE, 2025.

Yasaman Haghighi and Alexandre Alahi. Sencache: Accelerating diffusion model inference via sensitivity-aware caching. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 14295–14304, June 2026.

Jonathan Ho and Tim Salimans. Classifier-free diffusion guidance. *arXiv preprint arXiv:2207.12598*, 2022.

Team Hunyuan3D, Shuhui Yang, Mingxin Yang, Yifei Feng, Xin Huang, Sheng Zhang, Zebin He, Di Luo, Haolin Liu, Yunfei Zhao, et al. Hunyuan3d 2.1: From images to high-fidelity 3d assets with production-ready pbr material. *arXiv preprint arXiv:2506.15442*, 2025.

Akio Kodaira, Chenfeng Xu, Toshiki Hazama, Takanori Yoshimoto, Kohei Ohno, Shogo Mitsuhori, Soichi Sugano, Hanying Cho, Zhijian Liu, Masayoshi Tomizuka, et al. Streamdiffusion: A pipeline-level solution for real-time interactive generation. In *2025 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 12371–12380. IEEE, 2025.

Tuomas Kynkäänniemi, Miika Aittala, Tero Karras, Samuli Laine, Timo Aila, and Jaakko Lehtinen. Applying guidance in a limited interval improves sample and distribution quality in diffusion models. *Advances in Neural Information Processing Systems*, 37:122458–122483, 2024.

Zeqiang Lai, Yunfei Zhao, Zibo Zhao, Haolin Liu, Fuyun Wang, Huiwen Shi, Xianghui Yang, Qingxiang Lin, Jingwei Huang, Yuhong Liu, et al. Unleashing vecset diffusion model for fast shape generation. In *2025 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 2523–2533. IEEE, 2025.

Yunghee Lee, Byeonghyun Pak, Junwha Hong, and Hoseong Kim. Tortoise and hare guidance: Accelerating diffusion model inference with multirate integration. *Advances in Neural Information Processing Systems*, 38:2871–2898, 2025.

Mingkun Lei, Tong Zhao, Liangyu Yuan, and Chi Zhang. Budget-constrained step-level diffusion caching. In *Forty-third International Conference on Machine Learning*, 2026. URL https://openreview.net/forum?id=kN0NMoVWXG.

Haotang Li, Zhenyu Qi, Shaohan Henry Wang, Kebin Peng, Yutong Zhao, Zi Wang, Bo Liu, Huanrui Yang, and Sen He. Geocache: Training-free acceleration of multi-view texture diffusion via geometric delta transport. *arXiv preprint arXiv:2608.13255*, 2026.

Feng Liu, Shiwei Zhang, Xiaofeng Wang, Yujie Wei, Haonan Qiu, Yuzhong Zhao, Yingya Zhang, Qixiang Ye, and Fang Wan. Timestep embedding tells: It's time to cache for video diffusion model. In *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 7353–7363. IEEE, 2025a.

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

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

Zhengyao Lyu, Chenyang Si, Junhao Song, Zhenyu Yang, Yu Qiao, Ziwei Liu, and Kwan-Yee K Wong. Fastercache: Training-free video diffusion model acceleration with high quality. In *International Conference on Learning Representations*, volume 2025, pp. 33132–33156, 2025.

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

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

Stefan Stojanov, Anh Thai, and James M Rehg. Using shape to categorize: Low-shot learning with an explicit shape bias. In *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1798–1808. IEEE, 2021.

Ruitong Sun, Tianze Yang, Wei Niu, and Jin Sun. RSTR: Reducing spatiotemporal redundancy in diffusion transformers. In *Forty-third International Conference on Machine Learning*, 2026. URL https://openreview.net/forum?id=enfG9qcdz6.

Jiangshan Wang, Zeqiang Lai, Jiarui Chen, Jiayi Guo, Hang Guo, Xiu Li, Xiangyu Yue, and Chunchao Guo. Elastic diffusion transformer. In *Forty-third International Conference on Machine Learning*, 2026. URL https://openreview.net/forum?id=493YXJPdeq.

Shuang Wu, Youtian Lin, Feihu Zhang, Yifei Zeng, Jingxi Xu, Philip Torr, Xun Cao, and Yao Yao. Direct3d: Scalable image-to-3d generation via 3d latent diffusion transformer. *Advances in Neural Information Processing Systems*, 37:121859–121881, 2024.

Shuang Wu, Youtian Lin, Feihu Zhang, Yifei Zeng, Yikang Yang, Jiachen Qian, Siyu Zhu, Xun Cao, Philip Torr, Yao Yao, et al. Direct3d-s2: Gigascale 3d generation made easy with spatial sparse attention. *Advances in Neural Information Processing Systems*, 38:170778–170804, 2026.

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

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

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

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.

Hui Zhang, Tingwei Gao, Jie Shao, and Zuxuan Wu. Blockdance: Reuse structurally similar spatio-temporal features to accelerate diffusion transformers. In *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 12891–12900. IEEE, 2025.

Huixuan Zhang and Xiaojun Wan. How much to guide: Revisiting adaptive guidance in classifier-free guidance text-to-vision diffusion models. In *Proceedings of the 7th ACM International Conference on Multimedia in Asia*, pp. 1–8, 2025.

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

Tong Zhao, Mingkun Lei, Yucheng Han, and Chi Zhang. Bag: Budget-aware gating for diffusion caching. *arXiv preprint arXiv:2608.09231*, 2026.

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

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

Chang Zou, Xuyang Liu, Ting Liu, Siteng Huang, and Linfeng Zhang. Accelerating diffusion transformers with token-wise feature caching. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=yYZbZGo4ei.

A Motivation Diagnostic Protocols

Timing and temporal variation. Figure 1(a) reports ten exclusive-GPU runs per model, using Hunyuan3D with octree resolution 256 and a standalone 512 SLat model for TRELLIS.2. Bars and whiskers show mean per-run core-pipeline shares and sample standard deviations; CFG includes denoiser evaluations. Main-comparison speedups use separate generation timings after image preprocessing (Appendix C). Figure 1(b) uses ten separate Hunyuan3D trajectories with 29 valid adjacent-step comparisons per branch, excluding the initial self-comparison. Both heatmaps share a linear $1 - \cos(v_i, v_{i-1})$ scale (Section 3.1). These trajectories are unpaired with the endpoint cohort.

Endpoint treatments and accounting. The Hunyuan endpoint diagnostic is conditional on successful Full-CFG generation. The five treatments in Figure 3(a) share 50 object identifiers. All finite endpoint CD values, including the high-deviation tail, are retained. The objectwise comparison in Section 3.1 uses these shared IDs, rather than pairing positions on the sorted curves. CD uses the Full-CFG reference defined in Section 3.1. The documented fixed schedule is $T = 30$, $K = 10$, $r = 2$, with guidance scale $s = 5$; r denotes reused updates between fresh evaluations. This shared-ID comparison describes endpoint deviations on separate trajectories and does not establish a seed-matched causal effect. See Section 3.1 for its scope.

The eight fixed-treatment means use 50 finite CD records. The daggered Fixed CodaCache and separately measured CodaCache results are aggregate summaries excluded from the paired endpoint comparison. The $\mathrm{CD} > \tau_{\mathrm{CD}}$ column counts finite reference-deviation exceedances; it is distinct from recorded failed generations and is not a population failure rate.

Cutoff stress diagnostic. Figure 2(b) uses an intentionally selected ten-object cohort: five known Reuse-C threshold exceedances at $K = 10$ and five additional objects. This cohort probes stress cases rather than population failure prevalence. Seven treatments are displayed at $K \in \{0, 2, \ldots, 30\}$. Colors use finite-CD means on a logarithmic scale. The row-level records contain nine finite endpoints in each of five cells with a recorded failed generation and ten in each remaining cell. Crosses mark these failures and are excluded from finite-CD means; CD-threshold exceedances are not annotated. The archived $K = 30$ comparison has a nonzero residual CD even though no approximate update is applied, so we do not interpret it as a cutoff effect.

Local cache-error diagnostic. Figure 3(b) contains 130 aligned records, documented as repeated local events at $s = 5$, not 130 independent objects. The treatments are compared against fresh Full CFG at a common state under the archived diagnostic definition. Boxes show the median and interquartile range; whiskers extend to the most extreme observations within 1.5 interquartile ranges, and all event values are shown. The mean cosine similarity between the C- and U-reuse error vectors is 0.9259. These local events are unpaired with the endpoint cohort. Section 3.2 discusses the local-error interpretation and its limits for explaining endpoint deviation.

B CodaCache Implementation

This appendix details controller initialization and state updates, robust rate estimation and validity checks, and trace recording for the method in Section 4.

B.1 Causal action-selection loop

Table 4 lists the initialization and state updates needed to implement the equations in Section 4. The U-omission audit reuses the fresh C/U predictions from an executed F action; the C-reuse audit compares refreshed C with the old cache before overwrite. Both comparisons evaluate solver increments at the same pre-update latent and solver history. They require additional solver arithmetic but no extra denoiser evaluation, and their observations affect only subsequent decisions.

B.2 Action decomposition and failure boundary

Controller fallback applies only while the native latent and solver state remain finite. A non-finite native state invokes the generator's own failure handler rather than relabeling the update as a Full-CFG controller fallback.

At the same latent, timestep, and immutable solver history, the composed reuse-action discrepancy satisfies

$$\begin{aligned}\Delta_i(\bar{v}_{c,j}; h_i) - \Delta_i(v_{F,i}; h_i) = {} & \big[\Delta_i(\bar{v}_{c,j}; h_i) - \Delta_i(v_{c,i}; h_i)\big] \\ & + \big[\Delta_i(v_{c,i}; h_i) - \Delta_i(v_{F,i}; h_i)\big].\end{aligned} \tag{14}$$

Dividing the triangle-inequality bound by the shared ν_i yields the additive form in Eq. (5). Only the extrapolation from observed anchors is empirical.

Table 4: **Causal action-selection procedure for CodaCache.** All scores at update i are computed from completed history, and observations produced at i affect only subsequent decisions.

Step	Operation
Input	Pretrained denoiser f_θ, initial latent $z^{(0)}$, condition c, unconditional input $c_\emptyset$, native schedule $\{(t_i, \mathcal{S}_i)\}_{i=0}^{T-1}$, guidance scale s, and fixed controller configuration Θ.
Init.	Initialize the C cache, Full-CFG anchor, and both rate estimators as unavailable.
1	At update i, compute the trajectory distances and the two predicted score components from completed history. Unavailable or unsupported estimates are set to $+\infty$.
2	Select a_i by Eq. (6). Execute F, P, or R and construct $\widehat{v}_i$ according to Eq. (4).
3	If C is freshly evaluated and an old cache exists, compute the C-reuse audit y_i^C before overwriting the cache. A valid rate updates $L_{i+1}^{C,+}$. The fresh C prediction then becomes the new cache.
4	If $a_i = F$, compute q_i^U. A valid audit establishes the first Full-CFG anchor. Thereafter, update $L_{i+1}^{U,+}$ only if the rate to the previous anchor is valid, then replace the anchor with the valid current audit. Neither audit changes the action at update i.
5	Apply exactly one native solver update using $\widehat{v}_i$ and proceed to update $i+1$.

For the evaluated Euler solvers, $\Delta_i(v; h_i) = \delta t_i v$. With $\delta t_i \neq 0$ and $\epsilon = 0$, the common $|\delta t_i|$ cancels: the audits in Eqs. (8) and (12) become RMS prediction differences divided by $\|v_{c,i}\|_{\mathrm{RMS}}$. These experiments assess joint scoring, without isolating an advantage of increment-space normalization.

B.3 Online rate estimates and support checks

The one-sided rates in Eqs. (10) and (12) are tracked separately for $b \in \{U, C\}$. When a valid rate ℓ_i^b is observed, the update is

$$\mu_{i+1}^b = (1-\alpha)\mu_i^b + \alpha \ell_i^b, \qquad L_{i+1}^{b,+} = \mu_{i+1}^b + \kappa\, \mathrm{MAD}(\mathcal{W}_{i+1}^b), \tag{15}$$

where $\mathcal{W}_{i+1}^b$ contains the most recent W valid rates and MAD is their unscaled median absolute deviation; κ absorbs any desired scale factor. Each window begins empty with count zero. The first valid rate initializes μ^b directly, and a singleton window has zero MAD. Without a valid observation, the window, count, and estimator state are copied unchanged. All actions requiring component b remain infeasible until $n_{\min}$ valid rates for that component have been observed. We require $\epsilon, \nu_{\min}, D_{\min} > 0$, $\alpha \in (0, 1]$, $\kappa, \eta, \tau_{\mathrm{act}} \geq 0$, and $W, n_{\min} \in \{1, 2, \ldots\}$. The two support intervals and the mapping $\Lambda : t_i \mapsto \lambda_i$ are part of the fixed configuration reported in Table 5.

An audit is valid when its operands are finite and $\nu_i \geq \nu_{\min}$. A rate additionally requires a valid reference and $D \geq D_{\min}$. Only valid rates update the estimator window, count, and state. Valid U audits update the Full-CFG anchor even without a valid rate; invalid audits leave it unchanged. The controller uses separate support intervals for the U and C trajectory distances; leaving one interval invalidates only the affected score.

B.4 Fixed controller configuration

Table 5 lists the CodaCache controller constants shared by TRELLIS.2 and Hunyuan3D 2.1 across both datasets. Component ablations retain this configuration except for the changes specified in Table 3 and the ablation definitions below. The support intervals D^U and D^C are computed separately for each backbone using the same calibration rule and remain fixed throughout the reported runs.

The numerical constants ϵ, $\nu_{\min}$, and $D_{\min}$ prevent unstable normalization when the corresponding solver increments or trajectory distances are very small. The estimator parameters α, κ, and W control the update and robustness of the online rate estimates in Eq. (15). We use $n_{\min} = 2$, so an approximate action is considered only after the corresponding rate estimator has received enough valid observations.

The action threshold is fixed to $\tau_{\mathrm{act}} = 0.15$ for all evaluated settings. The time-distance weight is $\eta = 1$, giving the latent-motion and normalized time-separation terms equal nominal weight in Eq. (9).

Table 5: **CodaCache controller configuration.**

Parameter	Value / rule	Role
ϵ	10^{-6}	Numerical stabilizer
α	0.56	EMA update rate
κ	2.0	MAD multiplier
W	4	Recent-rate window size
$n_{\min}$	2	Minimum valid-rate count
η	1.0	Time-distance weight
$\nu_{\min}$	10^{-4}	Minimum valid increment scale
$D_{\min}$	10^{-4}	Minimum valid trajectory distance
τ_{act}	0.15	Action acceptance threshold
D^U	$[D_{\min},\ P_{95}(\mathcal{D}_U^{\text{cal}})]$	U-omission support
D^C	$[D_{\min},\ P_{95}(\mathcal{D}_C^{\text{cal}})]$	C-reuse support
Λ (TRELLIS.2)	$\lambda_i = t_i^{\text{rescaled}}$	Native Euler time
Λ (Hunyuan3D)	$\lambda_i = t_i^{\text{sched}}/N_{\text{train}}$	Normalized scheduler time
$(\lambda_{\min}, \lambda_{\max})$	$(0, 1)$	Complete-stage bounds

Here t_i^{rescaled} is TRELLIS.2's Euler time after `rescale_t=3.0`, before scaling for network input. In Hunyuan3D, t_i^{sched} denotes the scheduler timestep. We normalize it by the scheduler's `num_train_timesteps` parameter, $N_{\text{train}} = 1000$. The bounds cover each complete independent stage, including its terminal endpoint. Full-CFG calibration and online CodaCache decisions use these same mappings.

For calibration, each Full-CFG stage trajectory contributes all historical pairs $0 \le j < i < T$, where T is the number of solver updates in that stage.

We compute $D_{i|j}$ using Eq. (9) and retain pairs satisfying the component-specific validity checks in Appendix B.3. Pairs never cross inputs, runs, or cascade stages. For each backbone and component $b \in \{U, C\}$, $\mathcal{D}_b^{\text{cal}}$ pools valid distances across both evaluated datasets and calibration runs, with equal weight per valid pair.

TRELLIS.2 distances are constructed separately at the two cascade levels before pooling.

The support interval is $[D_{\min},\ P_{95}(\mathcal{D}_b^{\text{cal}})]$ and remains fixed throughout the reported runs. For $N_b = |\mathcal{D}_b^{\text{cal}}|$ sorted distances indexed from zero, P_{95} uses linear interpolation at rank $0.95(N_b - 1)$.

Cross-dataset calibration transfer. To test dependence on target-dataset calibration, we recompute the support bounds using Full-CFG trajectories from only the other dataset and keep all controller parameters fixed. Table 6 compares this transfer setting with the main pooled calibration.

All calibration variants use the same budget of $N_{\text{cal}} = 200$ inputs per backbone, drawn from distinct object IDs with one fixed view per object and evaluated with the same set of five random seeds (1,000 input–seed runs). Pooled calibration uses 100 inputs from Toys4K and 100 from Objaverse-XL; transfer uses 200 inputs exclusively from the non-target dataset. For Hunyuan3D 2.1, each input–seed run contributes one 30-step stage trajectory. For TRELLIS.2, each run contributes one eight-step trajectory at each cascade level; the levels are processed separately before pooling, as above.

Distance-term ablations use their reduced distance for rate denominators, online scores, and support checks. We recompute U/C P95 upper bounds on the same Full-CFG trajectories under the rules above, then fix them during inference. All other controller parameters, including initialization settings, remain unchanged.

Model weights remain fixed during calibration and evaluation. Each variant updates its online estimators from the sample's completed trajectory to select subsequent actions. Endpoint geometry, CD, and F-score are excluded from these decisions.

Separate acceptance configuration. In the Separate acceptance ablation, the R score in Eq. (5) is $\max(\widehat{R}_i^U, \widehat{R}_i^C)$ instead of the sum. Both component thresholds are fixed to $\tau_U = \tau_C = \tau_{\text{act}} = 0.15$

Table 6: **Cross-dataset support calibration.** In the main Pooled configuration, CodaCache's P95 support bounds are computed from valid distances pooled across both datasets' Full-CFG trajectories. Transfer uses only the other dataset for calibration. CD and F-score are valid-output means; FLOPs are relative to Full CFG, and Valid is the valid-output rate.

Backbone	Calibration	Evaluation	CD ↓	F-score ↑	FLOPs (%) ↓	Valid (%) ↑
TRELLIS.2	Pooled	Toys4K	0.0687	54.84	43.8	89.65
TRELLIS.2	Objaverse-XL only	Toys4K	0.0689	54.70	42.5	89.50
TRELLIS.2	Pooled	Objaverse-XL	0.0262	85.76	39.6	90.34
TRELLIS.2	Toys4K only	Objaverse-XL	0.0263	85.60	41.0	90.20
Hunyuan3D 2.1	Pooled	Toys4K	0.0159	75.35	44.7	92.57
Hunyuan3D 2.1	Objaverse-XL only	Toys4K	0.0160	75.20	43.5	92.40
Hunyuan3D 2.1	Pooled	Objaverse-XL	0.0590	64.62	37.6	93.83
Hunyuan3D 2.1	Toys4K only	Objaverse-XL	0.0593	64.30	39.2	93.60

across the four settings. The F/P scores, legal-action checks, and minimum-cost selection rule remain unchanged. This ablation uses the same initialization, distance definition, time mapping, per-backbone support intervals, estimator constants, and execution loop as CodaCache. Each variant updates its cache and estimators from its own completed trajectory, so their realized states and scores can differ.

B.5 Trace accounting and reproducibility

For each sample x, the action trace can be summarized by its first U-omission point and the lengths of consecutive C-reuse runs after refresh steps $j \in \{0, \dots, T-1\}$ with $a_j(x) \in \{F, P\}$:

$$\begin{aligned} i_{\mathrm{omit}}(x) &= \min\Big(\{i : a_i(x) \in \{P, R\}\} \cup \{T\}\Big), \\ m_j(x) &= \max\big\{m \in \{0, \dots, T-1-j\} : a_{j+1}(x) = \cdots = a_{j+m}(x) = R\big\}. \end{aligned} \tag{16}$$

The empty run $m = 0$ is always admissible; thus $m_j(x) = 0$ if the next action is not R or $j = T-1$. These are trace summaries, not controller constraints; F may recur after $i_{\mathrm{omit}}(x)$.

Every run records the selected action, both score components, trajectory distance, cache anchor, available audit observations, and measured controller overhead at each solver update. These records reproduce the reported action trace without post-hoc action relabeling.

C Experimental Protocol and Fixed-Policy Diagnostics

Pretrained backbones and numerical precision. The main comparison and adaptive component ablations use the official `microsoft/TRELLIS.2-4B` and `tencent/Hunyuan3D-2.1` releases (Xiang et al., 2026; Hunyuan3D et al., 2025). TRELLIS.2 uses the `1024_cascade` pipeline, with the released 512- and 1024-resolution shape models. Its shape denoisers use BF16, and the native shape decoder uses FP16. Hunyuan3D uses the `hunyuan3d-dit-v2-1` checkpoint in FP16, with `octree_resolution=384` for mesh extraction. All methods use the same backbone, resolution, and precision within each model–dataset setting.

Input images and preprocessing. We use all Toys4K objects. For Objaverse-XL, we randomly select 3,000 objects from 200 selected categories, including at least one object per category. Following Fast3Dcache (Yang et al., 2026), we render each object from 12 fixed viewpoints. Each view is processed as a separate single-image input. The generated mesh is evaluated against the corresponding object's ground-truth mesh. All rendered views are retained without quality filtering, giving 36,000 Objaverse-XL inputs per run. We follow the released image-conditioning pipelines of TRELLIS.2 and Hunyuan3D 2.1 (Xiang et al., 2026; Hunyuan3D et al., 2025). TRELLIS.2 uses available foreground alpha information or background removal, crops around the foreground, and composites it on a black background. Hunyuan3D 2.1 removes the background when needed, recenters the foreground on white, and applies the released resizing and normalization. Within each

model–dataset setting, Full CFG, CodaCache, and all baselines receive the same input image with the same backbone-specific preprocessing.

Sampling overrides. We retain each backbone's native first-order Euler solver and use guidance scales $s = 7.5$ for TRELLIS.2 and $s = 5.0$ for Hunyuan3D. TRELLIS.2 uses `rescale_t=3.0`; Hunyuan3D uses its native `FlowMatchEulerDiscreteScheduler` with 30 uniformly spaced input sigmas from zero to one. CodaCache starts independently at each TRELLIS.2 cascade level, resetting its cache, anchors, and rate estimates before initialization. The Full-CFG reference uses linear CFG at every update of the evaluated denoising stage. In TRELLIS.2, we set `guidance_rescale=0` and enable guidance over the full time interval, overriding the released guidance defaults. CodaCache selects $F/P/R$ actions on this same solver schedule.

Model and fixed-schedule configurations. Fixed-policy controls (Tables 2 and 8) operate independently at TRELLIS.2's 512- and 1024-resolution Shape SLat cascade levels, each with $(T, K, s) = (8, 3, 7.5)$. The caches and fixed schedule reset at the start of each level. For Hunyuan3D 2.1, we use $(T, K, s) = (30, 10, 5.0)$. With $r = 2$ by default, each fresh conditional prediction precedes up to two reused updates. We test reuse counts $r \in \{1, 2, 4\}$ on both backbones and additionally $r \in \{5, 6\}$ on Hunyuan3D. For TRELLIS.2, one refresh and four reuses cover all five suffix updates, so increasing r beyond four leaves the schedule unchanged. All solver updates are retained. The Full-CFG reference evaluates C and U at every update of each accelerated stage, and Step AG uses its conditional-score variant with fresh C after the cutoff.

Ablation controls. After the prefix, Reuse-C and Reuse-U hold the named prediction while refreshing the other; CacheBoth holds both predictions. Prune-C retains fresh U, and Reverse-CodaCache omits C and periodically reuses U. Last-Delta evaluates fresh C at each suffix update and reuses the most recently refreshed residual $\delta = v_c - v_u$ through $\widehat{v} = v_c + (s-1)\delta$. EMA-Delta uses the same rule with an exponential moving average of refreshed residuals ($\beta = 0.9$).

External baselines. Each baseline retains its original optimization scheme and scope when adapted to the image-to-3D backbones, with acceleration parameters from its paper or official recommended configuration. For Fast3Dcache (Yang et al., 2026), we use the reported default configuration $(\mu, \omega, \tau_{\text{cache}}, \xi, f_{\text{corr}}, \rho_a, \rho_{\text{CFG-OFF}}) = (-0.07, 0.7, 3, 0.7, 3, 0.2, 0.75)$.

FLOPs and generation time. FLOPs are measured only over the evaluated CFG denoising stages and normalized to Full CFG over the same stages within each model–dataset setting. End-to-end generation timing starts after image preprocessing and ends at generation completion. Speedups compare unrounded wall-clock times with Full CFG over this interval, with support intervals fixed. Offline Full-CFG trajectory generation and P95 support-bound computation are excluded.

Geometry evaluation. All methods use one common evaluator. Following the Fast3Dcache geometry protocol (Yang et al., 2026), each generated and ground-truth mesh is centered and isotropically scaled by its maximum extent. We uniformly sample 10,000 points from each surface and align the generated point cloud to ground truth using iterative closest point (ICP). Chamfer distance is the mean of the two directional, unsquared Euclidean nearest-neighbor distances. F-score is the harmonic mean of precision and recall at distance threshold $\tau_F = 0.05$ and is reported as a percentage. CD is compared only within a fixed model–dataset setting.

Evaluation scope and aggregation. Tables 1–3 use the object sets and all rendered views defined above, with shared identifiers and five end-to-end runs per method and dataset. Each sample is one input image. We exclude failures, empty meshes, and outputs with non-finite CD or F-score from the geometry means. We average each metric over valid outputs within each run, then average the five run means. Appendices E and F state the scopes of the remaining diagnostics. Across the five runs, Full CFG/CodaCache valid-output rates are $89.69/89.65\%$ (TRELLIS.2/Toys4K), $90.45/90.34\%$ (TRELLIS.2/Objaverse-XL), $92.66/92.57\%$ (Hunyuan3D 2.1/Toys4K), and $94.23/93.83\%$ (Hunyuan3D 2.1/Objaverse-XL). Valid outputs are nonempty meshes with finite CD and F-score; the maximum gap is 0.40 percentage points. On TRELLIS.2/Objaverse-XL, unrounded CD means of 0.02614 (Full CFG) and 0.02618 (CodaCache) differ by 0.153%, reported as 0.15%. For Full CFG and CodaCache, we compute five-run standard deviations within each method–setting group. We

express the CD and FLOPs standard deviations as percentages of their corresponding means, and the F-score standard deviation in percentage points. Arithmetic means across the eight groups are 0.35%, 4%, and 0.45 percentage points, respectively. These describe run-to-run variability.

D ABSOLUTE END-TO-END THROUGHPUT

Tables 7 and 8 report measured throughput for the main/component and fixed-policy comparisons, respectively, using the timing protocol in Appendix C.

Table 7: End-to-end throughput (assets/s, higher is better) for Tables 1 and 3. Bold/underline mark the best/second-best unrounded values; Hunyuan is version 2.1.

	Toys4K		Objaverse-XL	
Method	TRELLIS.2	Hunyuan3D	TRELLIS.2	Hunyuan3D
Main comparison				
Full CFG	0.5055	0.1873	0.3147	0.2394
Step AG	<u>1.1070</u>	<u>0.4083</u>	0.5853	<u>0.5051</u>
Adaptive Guidance	0.8695	0.2716	0.6168	0.3471
Limited-Interval	0.9503	0.3315	0.6672	0.4477
THG	0.6925	0.3240	0.5570	0.2729
TCFG	0.9807	0.3390	0.6546	0.3902
StreamDiff. R-CFG	0.7330	0.4008	0.6200	0.4860
RSTR	1.0161	0.3596	0.6042	0.3304
Fast3Dcache	0.5850	0.3128	0.3651	0.2969
SenCache	0.8391	0.3858	0.4846	0.4118
ResCa	0.9150	0.3147	0.6074	0.3735
DPCache	1.0009	0.3652	<u>0.6703</u>	0.4022
CodaCache	**1.2385**	**0.4383**	**0.8591**	**0.6200**
Adaptive controller ablation				
Full CFG	0.5055	0.1873	0.3147	0.2394
Separate acceptance	0.9813	0.3323	<u>0.7367</u>	<u>0.5849</u>
w/o R (F/P only)	0.6218	0.2603	0.3808	0.2992
w/o P (F/R only)	0.8189	0.3446	0.4689	0.3543
w/o latent drift	0.9150	0.3727	0.5444	0.4237
w/o time distance	<u>1.0161</u>	<u>0.3971</u>	0.7112	0.4860
CodaCache	**1.2385**	**0.4383**	**0.8591**	**0.6200**

Table 8: End-to-end throughput (assets/s, higher is better) for selected fixed-policy controls. Bold/underline mark the best/second-best distinct unrounded values in each setting.

	Toys4K		Objaverse-XL	
Method	TRELLIS.2	Hunyuan3D	TRELLIS.2	Hunyuan3D
Full CFG	0.5055	0.1873	0.3147	0.2394
Reuse-C	0.7500	0.2890	0.4699	0.3638
Reuse-U	0.7500	0.2890	0.4699	0.3638
CacheBoth	1.1476	0.4216	0.7237	0.5524
Step AG	1.1070	0.4083	0.5853	0.5051
Fixed CodaCache-r1	<u>1.1913</u>	<u>0.4310</u>	0.7856	<u>0.5891</u>
Fixed CodaCache-r2	1.0385	0.3683	<u>0.7991</u>	0.5600
Fixed CodaCache-r4	**1.2895**	**0.4458**	**0.9478**	**0.6544**
Prune-C	1.1070	0.4083	0.5853	0.5051
Reverse-CodaCache	1.1385	0.4183	<u>0.7991</u>	0.5800
Last-Delta	0.7500	0.2890	0.4699	0.3638
EMA-Delta	0.7500	0.2890	0.4699	0.3638

E PARAMETER SENSITIVITY

Table 9 evaluates parameter sensitivity on Hunyuan3D 2.1 with Toys4K. We vary one parameter at a time, fixing $n_{\min} = 2$ for the threshold sweep and $\tau_{\text{act}} = 0.15$ for the history-count sweep. The remaining controller configuration follows Table 5.

Table 9: **Parameter sensitivity on Hunyuan3D 2.1 with Toys4K.** CD, F-score, and measured FLOPs follow the definitions in Appendix C. FLOPs are expressed as a percentage of Full CFG. Bold and underlining mark the best and second-best distinct values within each sweep. The dagger marks the shared default configuration.

Parameter value	CD ↓	F-score (%) ↑	FLOPs (%) ↓
Action threshold τ_{act}*;* $n_{\min} = 2$			
0.05	**0.0152**	<u>75.34</u>	64.1
0.10	<u>0.0158</u>	75.33	52.9
$0.15^{\dagger}$	0.0159	**75.35**	44.7
0.20	0.0234	73.27	<u>41.2</u>
0.30	0.0328	70.68	**35.8**
Minimum valid-rate count $n_{\min}$*;* $\tau_{\text{act}} = 0.15$			
1	<u>0.0337</u>	71.20	**37.3**
$2^{\dagger}$	**0.0159**	<u>75.35</u>	<u>44.7</u>
3	**0.0159**	**75.36**	52.1

Action threshold. Increasing τ_{act} reduced measured FLOPs across the tested values. From 0.10 to 0.15, FLOPs decreased by 8.2 percentage points, while CD changed from 0.0158 to 0.0159 and F-score changed by 0.02 percentage points. Raising the threshold from 0.15 to 0.20 saved a further 3.5 percentage points of FLOPs, but CD increased by 47.2% and F-score decreased by 2.08 percentage points. The two largest tested thresholds therefore reduced geometric quality in this setting.

Minimum valid-rate count. At $n_{\min} = 1$, CD was more than twice its value at $n_{\min} = 2$, and F-score was 4.15 percentage points lower. Increasing $n_{\min}$ from 2 to 3 added 7.4 percentage points of FLOPs. CD was unchanged at the reported precision, while F-score increased by 0.01 percentage points. Together, the sweeps support $(\tau_{\text{act}}, n_{\min}) = (0.15, 2)$ as an empirical efficiency–quality compromise for this model and dataset.

F HISTORY-BASED SCORE AND ACTION DIAGNOSTICS

Diagnostic scope. We compare CodaCache and Separate acceptance on 100 Toys4K inputs using Hunyuan3D 2.1, with three paired seeds and 30 updates per trajectory. Both methods share inputs and initial noise but maintain their own latent trajectories, caches, and rate estimates. Each method contributes 300 trajectories and 9000 updates. Both acceptance thresholds are 0.15 for Separate acceptance, matching CodaCache's τ_{act}.

Same-state measurements. Independent replay evaluates fresh C and U at each saved decision state, using its pre-update solver history and C cache. Let $d_F = \Delta_i(v_{F,i}; h_i)$, $d_C = \Delta_i(v_{c,i}; h_i)$, and $d_{\text{old}} = \Delta_i(\bar{v}_{c,j}; h_i)$. All candidates use the same state and solver history. With the common normalization ν_i from Eq. (7), we measure

$$e_i^U = \frac{\|d_F - d_C\|_{\text{RMS}}}{\nu_i}, \qquad e_i^C = \frac{\|d_C - d_{\text{old}}\|_{\text{RMS}}}{\nu_i}, \qquad e_i^R = \frac{\|d_F - d_{\text{old}}\|_{\text{RMS}}}{\nu_i}.$$

The error for P is e_i^U. The measured R error is the norm of the combined increment difference, rather than $e_i^U + e_i^C$. Replay evaluations do not update the original trajectory, cache, or estimators, and are excluded from the main timing and FLOPs measurements.

Aggregation. Score diagnostics pair each finite, supported pre-decision score with its valid measured component. Underestimation means $e_i^b > \widehat{R}_i^b$ for $b \in \{U, C\}$. Accepted-action statistics use

the valid measured errors for the actions actually selected on each method's trajectory. We compute medians and P95 by pooling all valid steps within each reported group. These are descriptive summaries of pooled steps, which include correlated observations from the same trajectory.

Table 10: **History-based scores and accepted-action errors.** Spearman ρ compares history-based scores with measured local components. Underestimation and threshold exceedance are distinct events. All accepted P/R steps have valid diagnostic errors in this cohort. Counts refer to steps, and $\tau = 0.15$.

(a) History-based component scores

Method	Component	Valid N	Spearman ρ	Underestimated N	Rate (%)
CodaCache	U	7979	0.514	161	2.02
CodaCache	C	8100	0.711	1890	23.33
Separate acceptance	U	7978	0.515	160	2.01
Separate acceptance	C	8088	0.753	2204	27.25

(b) Actually accepted actions

Method	Action	Total/valid N	Median error	P95 error	$N(e > \tau)$
CodaCache	P	3658/3658	0.0470	0.1017	0
CodaCache	R	2525/2525	0.0516	0.0961	0
Separate acceptance	P	706/706	0.0317	0.0892	0
Separate acceptance	R	5470/5470	0.0733	0.1503	275

Table 11: **Same-state rule comparison and realized action counts.** In (a), N is the total, valid-R, and paired-P/R count; these coincide in every group. The final column is the median of paired differences $e_i^U - e_i^R$, not a difference of group medians. In (b), post-initialization counts exclude the initial F observations. In (c), the three F reasons are mutually exclusive; C unavailability overlaps these reasons and only prevents R.

(a) Same-state rule comparison

Trajectory source	Group	N	Median e^R	P95 e^R	$N(e^R > 0.15)$	Median $(e^U - e^R)$
CodaCache	Common	2525	0.0516	0.0961	0	-0.0105
CodaCache	Disagreement	3652	0.0682	0.1202	21	-0.0141
Separate acceptance	Common	1914	0.0551	0.0988	0	-0.0111
Separate acceptance	Disagreement	3556	0.0882	0.1645	275	-0.0309

(b) Realized actions

Method	Initial F	Later F	Later P	Later R	Total
CodaCache	900	1917	3658	2525	9000
Separate acceptance	900	1924	706	5470	9000

(c) Post-initialization fallback reasons

	F reason			C unavailable		
Method	U unavailable	U $> \tau$	Other	Total	To P	To F
CodaCache	121	1796	0	0	0	0
Separate acceptance	122	1802	0	12	0	12

Score interpretation. Both policies show positive rank correlations between history-based scores and measured components (Table 10(a)). C-component underestimation remains appreciable for both policies, so the history-based scores are not guaranteed upper bounds. Separate acceptance has slightly higher correlations despite its larger accepted-R error tail (Table 10(b)). Thus correlation alone does not establish the reliability of an acceptance rule.

Comparing rules at the same state. For each trajectory source, we restrict the comparison to steps with a valid C cache and two finite, supported scores. The common group satisfies $\widehat{R}_i^U + \widehat{R}_i^C \leq 0.15$. The disagreement group satisfies $\max(\widehat{R}_i^U, \widehat{R}_i^C) \leq 0.15 < \widehat{R}_i^U + \widehat{R}_i^C$. At these disagreement

states, CodaCache's rule selects P and Separate acceptance selects R. Table 11(a) evaluates both candidate errors on each saved state, keeping the two trajectory sources separate.

The disagreement groups have larger R-error P95 values than the common groups on both trajectory sources. The negative medians of the paired error differences favor C refresh over C reuse at these disagreement states. All 275 threshold exceedances among Separate acceptance's accepted R steps occur in its disagreement group. These same-state comparisons support the joint rule's local selectivity in this diagnostic, without establishing a causal effect on complete-trajectory geometry.

Action composition. Separate acceptance executes more R and fewer P actions, while the two policies have similar F counts in this cohort. These counts characterize action selection within the diagnostic subset. The main-evaluation FLOPs in Table 3 are measured independently; the subset counts do not establish the cause of that FLOPs difference.