SOURCEMASK: PROVENANCE-AWARE SOLVER CONTROL FOR DIFFUSION CACHING

Anonymous authors
Paper under double-blind review

ABSTRACT

Diffusion caching replaces expensive network evaluations with reused or predicted values. Existing cache policies determine when to refresh and what to return, while the numerical sampler usually consumes fresh and approximate evaluations in the same way. We introduce SOURCEMASK, a provenance-aware interface for UniPC: the current corrector is enabled when both classifier-free guidance branches are fresh, and every returned evaluation is retained in predictor history. This rule adds no denoiser call. A complete-step analysis identifies how current-corrector and predictor-history paths jointly transmit cache error. On 944 Wan video prompts, SOURCEMASK reduces LPIPS to the matched full-compute trajectory by 6.16%, with improvements on 935 prompts at the same evaluation budget. Separate studies establish gains for direct reuse, output prediction, and held-out SeaCache/SenCache combinations. A predictor-matched extension further controls UniP's high-order residual using nested-history agreement. Experiments across solver orders, sampling grids, and a second video model characterize where these controls help. The results establish evaluation provenance as a practical signal for coordinating diffusion caches with predictor–corrector solvers.

1 INTRODUCTION

Video diffusion inference repeatedly evaluates a large denoiser to integrate a generative trajectory. Training-free caches reduce this cost by reusing intermediate features or forecasting network outputs across timesteps (Ma et al., 2024; Lv et al., 2025; Liu et al., 2025a; Kahatapitiya et al., 2025). Their design centers on two questions: when should an evaluation be refreshed, and what should a cache hit return? We study a third decision at the cache–solver boundary: how should the sampler use an evaluation once it knows that the value is approximate?

This decision is consequential for predictor–corrector sampling. In UniPC (Zhao et al., 2023), the current model output influences both the UniC corrector and the history used by UniP. These paths serve different purposes. An approximate evaluation can remain useful for predicting the trajectory while being unsuitable for the current correction. Yet the standard interface exposes only a numerical tensor, hiding whether it came from a fresh denoiser call, feature reuse, or output prediction. The cache already knows this information, so exposing it requires no estimate of an unavailable fresh output.

We introduce SOURCEMASK, a branch-aware source contract for UniPC. Each guidance branch returns its output and an execution-source label. The current UniC is enabled only when both branches are fresh; all outputs continue to enter native predictor history. The same interface supports direct reuse, predictive output caches, and dynamic refresh controllers. For predictive caches, we additionally study PW-UNIP, which uses disagreement between nested paid-fresh-history fits to scale the high-order predictor residual. We call the combined variant ORDERCACHE.

Our analysis treats the complete solver step, from current correction through history insertion to the next prediction. It separates the cache-error transmission removed by masking from the fresh correction that is sacrificed. This tradeoff explains why the effect depends on the sampling grid and cache mechanism, and motivates controls on gate choice, correction location, and fixed-action replay.

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.

The empirical study follows the same progression. A 944-prompt evaluation establishes the core gain; separate fixed-budget and held-out controller studies test composition; branch, solver-order, and sampling-step experiments identify the mechanism and its operating range. The main result is improved fidelity to a matched full-compute trajectory. Taken together, the study contributes a minimal cache–solver interface, a complete-step explanation of its effect, and a systematic evaluation of provenance-aware solver control.

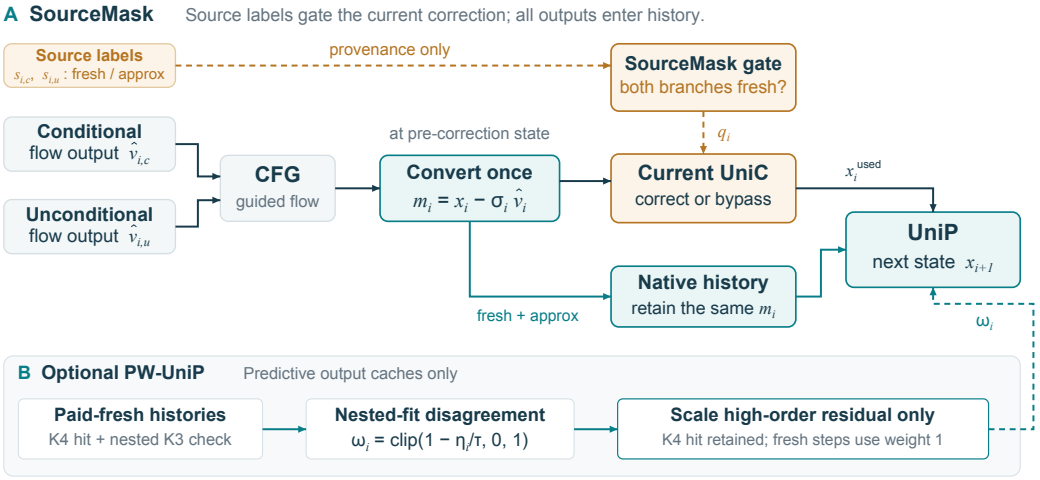


Figure 1: SourceMask separates provenance from numerical values. Source labels control only the current UniC; the same pre-correction data estimate always enters native history. Optional PW-UniP uses nested paid-fresh-history predictions to scale only UniP's high-order residual.

2 Related Work

Refresh decisions and cache values. DeepCache reuses intermediate features; PAB and FasterCache exploit attention and guidance redundancy (Ma et al., 2024; Zhao et al., 2025; Lv et al., 2025). Adaptive controllers choose refreshes from temporal, motion, spectral, or sensitivity signals (Liu et al., 2025a; Kahatapitiya et al., 2025; Chung et al., 2026; Haghighi & Alahi, 2026). EpaCache calibrates downstream reuse impact to allocate thresholds (Liu et al., 2026). A complementary line improves the returned approximation: TaylorSeer forecasts features, DiCache aligns cache trajectories, and FoCa/HyCa model feature evolution with numerical predictors (Liu et al., 2025b; Bu et al., 2026; Zheng et al., 2025; 2026). SourceMask consumes the source decisions made by these cache mechanisms at the outer solver interface.

Numerical sampling with approximate evaluations. DDIM, DPM-Solver, and UniPC accelerate generation through improved trajectory integration (Song et al., 2021; Lu et al., 2022a;b; Zhao et al., 2023). The closest solver-side interventions address different quantities: DC-Solver compensates the mismatch between a corrected state and its available evaluation through history modification (Zhao et al., 2024); ERTACache rectifies cache error and adjusts integration intervals (Peng et al., 2026). Our intervention directly maps observed execution provenance to the current UniC action. It preserves the cache-value construction, native history insertion, and time grid. Appendix G gives a detailed operational comparison.

3 Provenance-Aware Solver Control

3.1 The Cache–Solver Interface

At step i, let x_i be the pre-correction latent and $v_{i,c}, v_{i,u}$ the fresh conditional and unconditional flow outputs. Classifier-free guidance (CFG) with scale w gives

$$v_i = v_{i,u} + w(v_{i,c} - v_{i,u}). \tag{1}$$

In the studied FlowUniPC configuration, $\alpha_i = 1 - \sigma_i$, and the scheduler converts a flow output to a data estimate

$$m_i(v) = x_i - \sigma_i v. \tag{2}$$

The execution order is essential: compute m_i at x_i, optionally correct x_i with UniC, insert that same m_i into history, then apply UniP from the resulting state. There is no network re-evaluation after correction. Thus the current approximation reaches the next latent through both the corrector and predictor-history paths in Figure 1.

For analysis at an identical incoming state and history, write a cache-produced branch output as $\widetilde{v}_{i,b} = v_{i,b} + e_{i,b}$. Its guided error is

$$\widetilde{v}_i = v_i + e_i, \qquad e_i = (1-w)e_{i,u} + we_{i,c}. \tag{3}$$

The runtime interface exposes the execution source rather than e_i, whose measurement requires a fresh denoiser call. Approximation error at a fixed state is also distinct from later rollout drift: once a solver action changes x_{i+1}, future outputs and dynamic cache decisions may change.

3.2 SOURCEMASK

Each cache adapter returns $(\widehat{v}_{i,b}, s_{i,b})$, where $b \in \{c, u\}$ and $s_{i,b} \in \{\mathsf{fresh}, \mathsf{approx}\}$. The label describes the returned evaluation: a branch is approximate if a reused or predicted component contributes to its output, even if other components are recomputed. We define

$$q_i = \mathbf{1}[s_{i,c} = \mathsf{fresh} \ \wedge \ s_{i,u} = \mathsf{fresh}]. \tag{4}$$

SOURCEMASK enables the current UniC when $q_i = 1$ and bypasses it otherwise. Every converted output enters native history and UniP. The AND rule supplies a simple source contract for CFG, where either branch can introduce approximation error. Section 5.4 evaluates alternative branch aggregations.

The implementation changes one scheduler decision per step and adds neither a denoiser call nor a persistent tensor cache. In the native scheduler, the corrector at step $i > 0$ is addressed by disable index $i - 1$; the first step has no preceding correction. Appendix B gives the full algorithm. A fixed-action experiment holds the branch fresh/hit schedule constant, whereas an online controller produces source labels from each candidate's evolving trajectory. In both cases, candidates retain their own latents, cache anchors, and solver histories.

3.3 PREDICTOR-MATCHED HIGH-ORDER CONTROL

Predictive output caches expose an additional consistency signal. Our OutputRidge adapter fits K4 to the four most recent paid-fresh raw flow outputs and independently fits a nested K3 companion to the most recent three. K4 returns the hit value; K3 assesses its stability. Conditional and unconditional histories are separate, and only fresh outputs enter these fitting histories. Native UniPC history, in contrast, includes both fresh and predicted evaluations.

Let $p^b_{4,i}$ and $p^b_{3,i}$ denote the two branch predictions. Their normalized branch-product disagreement is

$$\eta_i = \frac{\sqrt{w^2\|p^c_{4,i} - p^c_{3,i}\|_F^2 + (1-w)^2\|p^u_{4,i} - p^u_{3,i}\|_F^2}}{\sqrt{w^2\|p^c_{4,i}\|_F^2 + (1-w)^2\|p^u_{4,i}\|_F^2} + 10^{-12}}. \tag{5}$$

Computing the norm before combining branches avoids cancellation between conditional and unconditional disagreement. At a predictive hit, we set

$$\omega_i = \mathrm{clip}(1 - \eta_i/\tau, 0, 1), \qquad \tau = 0.005, \tag{6}$$

and multiply only UniP's high-order residual by ω_i. The base update, returned K4 value, and CFG combination are preserved; fresh steps use $\omega_i = 1$. The combined SOURCEMASK +PW-UNIP policy is ORDERCACHE. Appendix D gives the exact fits and update. This signal is matched to the predictor producing the hit, which we test directly with cross-mechanism controls and a threshold sweep.

To make the intervention explicit, write the native predictor as a base update plus a high-order residual:

$$x_{i+1} = \underbrace{A_i x_i^{\text{used}} - \alpha_{i+1}\phi_i^P \widehat{m}_i}_{\text{base update}} + \omega_i \underbrace{\left(-\alpha_{i+1} B_i^P \sum_{j=1}^{K_i^P - 1} \rho_j^P \frac{m_{i-j} - \widehat{m}_i}{r_j} \right)}_{H_i\text{: high-order residual}}. \tag{7}$$

Here x_i^{used} is the state selected by the source gate, $A_i = \sigma_{i+1}/\sigma_i$, and K_i^P is the realized predictor order. For the native `bh2` configuration, $\lambda_i = \log \alpha_i - \log \sigma_i$, $h_i^P = \lambda_{i+1} - \lambda_i$, $\phi_i^P = B_i^P = \text{expm1}(-h_i^P)$, and $r_j = (\lambda_{i-j} - \lambda_i)/h_i^P$; ρ_j^P are the native UniPC coefficients. At first order the sum is empty. Thus PW-UniP continuously interpolates between the base update and the native higher-order predictor, without substituting K3 for K4 or changing the sampling grid.

4 A Complete-Step View of Cache Error

Fix the incoming latent, full solver history, and time grid of the deterministic second-order FlowUniPC configuration. With these quantities fixed, the complete UniC–history–UniP step is affine in the current flow evaluation:

$$T_{\text{on}}(v) = a_{\text{on}} + b_{\text{on}} v, \qquad T_{\text{off}}(v) = a_{\text{off}} + b_{\text{off}} v. \tag{8}$$

The coefficients include both the current correction and subsequent predictor. Let $d_i = T_{\text{off}}(v_i) - T_{\text{on}}(v_i)$ be the fresh-correction displacement lost when UniC is bypassed. Relative to the fresh-on same-state reference, ordinary caching has error $b_{\text{on}} e_i$, while SourceMask has error $d_i + b_{\text{off}} e_i$. The squared-error gain is therefore

$$G_i = (b_{\text{on}}^2 - b_{\text{off}}^2)\|e_i\|_2^2 - \|d_i\|_2^2 - 2 b_{\text{off}} \langle d_i, e_i \rangle. \tag{9}$$

Positive G_i favors masking. The first term captures the changed transmission of cache error; the remaining terms account for the lost correction and its alignment with that error. Provenance identifies the steps at which this tradeoff arises without requiring a fresh-output oracle. Its benefit depends on solver coefficients, error magnitude, and error direction. Appendix C derives the coefficients and verifies the identity against the native implementation.

The two paths can be read directly from the coefficients. Let k_i^C be the current UniC response to a flow perturbation and collect the current data-estimate coefficient in the predictor as β_i. Then

$$b_{\text{off}} = -\sigma_i \beta_i, \qquad b_{\text{on}} = b_{\text{off}} + A_i k_i^C, \qquad d_i = A_i [x_i - x_i^C(v_i)]. \tag{10}$$

The term $-\sigma_i \beta_i$ remains because the approximate output is inserted into history even when the correction is skipped. Masking removes $A_i k_i^C$, rather than removing all dependence on cache error. This distinction is the reason to analyze the completed update instead of treating UniC in isolation. It also makes the rule complementary to better cache values: reducing e_i and changing its transmission act on different parts of Eq. (9).

Two consequences guide the experiments. First, the error immediately after UniC is insufficient to determine the completed-step effect, because the same evaluation also enters UniP. Second, with zero terminal noise and a first-order final predictor, the final update depends on the pre-correction m_{N-1} and is unchanged by a current UniC toggle. This yields an exact terminal zero-effect control.

For a fixed set S of approximate-source steps, SourceMask is also equivalent to the compiled native time mask

$$\mathcal{D}(S) = \{i - 1 : i \in S,\ i > 0\}. \tag{11}$$

The two policies execute identical actions and trajectories by induction. Online provenance supplies this mask separately for each realized source trace. We test this distinction with transferred and target-selected time masks. The analysis concerns a complete step at fixed incoming state and history; the empirical rollouts measure its accumulated effect.

5 EXPERIMENTS

5.1 EVALUATION PROTOCOL

Our primary model is Wan2.1-T2V-1.3B with its native UniPC sampler (Wan Team, 2025; Zhao et al., 2023): 10 steps, shift 5, CFG 5, and 81 frames at 832×480. We evaluate direct output reuse (DenoiserReuse), K4 output prediction (OutputRidge), and the dynamic SeaCache/SenCache controllers. OutputRidge uses the fixed hit set $\{5, 7, 9\}$. Each solver candidate starts from the same prompt and initial noise, then maintains its own states and histories. Fixed-budget comparisons share cache rules, predictor coefficients, and branch action schedules. The 944-prompt suite uses one seed; the 16-prompt studies use two, averaged within each prompt. Held-out controller tests use 50 steps.

We measure LPIPS-to-Full, SSIM, and PSNR against the paired full-compute trajectory on the same model, grid, and solver order. These metrics quantify trajectory fidelity. For N steps, branch-level NFE is $n_c + n_u$ and cache ratio is $1 - \mathrm{NFE}/(2N)$, where n_c, n_u count fresh branch evaluations. Reported relative reductions are

$$R_{\mathrm{LPIPS}} = 100 \left(1 - \frac{\overline{L}_{\mathrm{SM}}}{\overline{L}_{\mathrm{ordinary}}}\right) \%, \tag{12}$$

with equal prompt weight; confidence intervals resample prompt clusters. Denoiser time and warm end-to-end time are identified separately. Appendix A specifies all splits, metrics, schedules, and compute accounting.

5.2 FIDELITY AT A FIXED EVALUATION BUDGET

On the 944-prompt VBench suite, SOURCEMASK reduces OutputRidge LPIPS from 0.0899 to 0.0843, a 6.16% improvement with 935/944 prompt wins (Table 1). SSIM and PSNR improve as well. All cache treatments use the same predictor construction and evaluation schedule, with denoiser time near 25.3 s versus 36.1 s for Full-10. Thus provenance control improves the fidelity of the accelerated trajectory at the existing cache budget. Adding PW-UNIP yields a further 0.40% LPIPS reduction, separating the core source effect from the smaller large-scale extension gain.

Table 1: 944-prompt evaluation on Wan (one seed). Cache variants share the K4 construction and denoiser schedule. Metrics compare with Full-10; time is denoiser time per video.

Solver treatment	LPIPS ↓	SSIM ↑	PSNR ↑	Time (s)
Full-10	0.0000	1.0000	–	36.073
OutputRidge, ordinary	0.0899	0.8649	26.802	25.279
+ SOURCEMASK	0.0843	0.8706	27.234	25.265
+ PW-UNIP (ORDERCACHE)	**0.0840**	**0.8746**	**27.658**	25.248

The independent p44–p59 split confirms the gain for both direct reuse and prediction (Table 2). All 16 prompts improve at each of the three operating points, with LPIPS reductions of 4.23–8.50%. At the mid reuse budget, SOURCEMASK also improves over a seven-step full-compute control using the same number of fresh evaluations: 0.1435 versus 0.1511 LPIPS. Recorded warm end-to-end time is 30.607 s for reuse+SOURCEMASK, 30.584 s for ordinary reuse, and 41.578 s for Full-10. The cache supplies the speedup; the gate improves fidelity at that operating point.

Table 2: Fixed-action comparisons on 16 prompts and two seeds. NFE counts fresh CFG-branch calls (Full-10: 20); confidence intervals describe relative LPIPS reduction.

Cache	NFE	Ordinary	SOURCEMASK	Wins	Reduction (95% CI)
DenoiserReuse-mid	14	0.1568	0.1435	16/16	8.50% [7.29, 9.84]
DenoiserReuse-fast	12	0.2423	0.2243	16/16	7.42% [6.11, 8.63]
OutputRidge	14	0.1201	0.1150	16/16	4.23% [3.19, 5.45]

A six-budget reuse sweep extends the comparison from four to nine fresh steps. SOURCEMASK improves LPIPS at each non-degenerate f4–f8 point, with reductions from 2.41% to 11.62%. At f9, the only hit is terminal and the two outputs coincide, matching the complete-step analysis. Appendix F gives the full sweep; Figure 2 shows a paired speed–fidelity view.

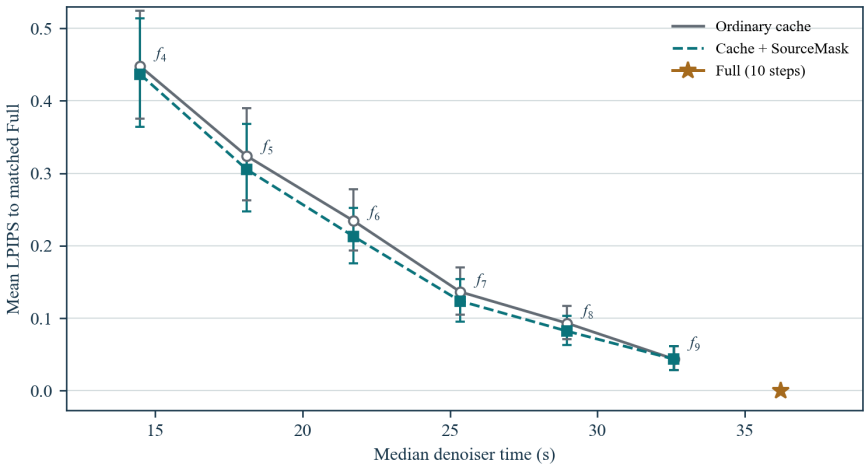


Figure 2: Paired speed–fidelity comparison on p68–p83, seed 0. Each pair uses the same reuse schedule; f_k denotes k fresh steps out of ten. Axes report median denoiser time and mean LPIPS; intervals are 95% prompt-bootstrap intervals.

5.3 COMPOSITION WITH DYNAMIC CACHE CONTROLLERS

The held-out 50-step studies test spectral and sensitivity-based refresh policies on p68–p83, two seeds. Table 3 summarizes all four operating points. SOURCEMASK reduces SeaCache LPIPS by 16.90% at $\delta = 0.20$ and 3.45% at $\delta = 0.35$. It improves the official-add SenCache adapter by 3.17% at both locally adapted settings, with 15/16 prompt wins each. SSIM and PSNR also improve in every pair; complete metrics and timing appear in Appendix F.

Table 3: Held-out controller composition: 16 prompts, two seeds, 50 steps. Each pair has equal realized NFE in every prompt–seed cell. SenCache uses the official out-of-place residual addition at local settings $m \in \{1, 2\}$.

Controller	Mean NFE	Ordinary	SOURCEMASK	Wins	Reduction (95% CI)
SeaCache $\delta = 0.20$	47.94	0.0823	0.0684	16/16	16.90% [11.88, 22.69]
SeaCache $\delta = 0.35$	34.00	0.1814	0.1752	12/16	3.45% [1.22, 5.41]
SenCache $m = 1$	36.81	0.1759	0.1703	15/16	3.17% [1.98, 4.37]
SenCache $m = 2$	36.00	0.1769	0.1713	15/16	3.17% [1.99, 4.35]

The trace audit distinguishes equal budgets from equal actions. SeaCache-0.20 changes hit locations in two of 32 paired cells while retaining their evaluation counts; SeaCache-0.35 and both SenCache settings retain identical source traces in every cell. These measurements show that the interface composes with online controllers and identify where trajectory feedback changes their decisions. The earlier eight-prompt development study is retained in Appendix F.6.

5.4 WHAT DOES THE SOURCE GATE CONTROL?

Branch aggregation. An asymmetric CFG-hit schedule makes six source gates distinguishable at the same 15-branch-NFE budget (Table 4). The default AND gate reduces LPIPS by 4.66% and improves every prompt. Conditional-only is slightly stronger, with 0.24% lower LPIPS than

AND (95% CI [0.07, 0.43]); the two differ only at the unconditional-only hit at step 5. OR and unconditional-only slightly worsen LPIPS. Provenance therefore matters at branch resolution, while the best aggregation can depend on the branch schedule. AND supplies a consistent contract for both branches.

Table 4: Branch gate ablation, p44–p59, two seeds, 15 branch NFEs. q_c, q_u indicate fresh conditional/unconditional evaluations; q enables current UniC.

Gate	q	LPIPS ↓	Reduction (%)	Wins
Always enable	1	0.5392	0.00	–
AND (SOURCEMASK)	$q_c \wedge q_u$	0.5140	4.66	16/16
Conditional only	q_c	**0.5128**	**4.90**	16/16
Unconditional only	q_u	0.5409	−0.32	3/16
OR	$q_c \vee q_u$	0.5411	−0.35	1/16
Always disable	0	0.5278	2.11	9/16

Always disabling UniC also improves the mean, but by 2.11% with only 9/16 prompt wins, compared with 4.66% and 16/16 for the source-conditioned AND gate. The contrast shows why the fresh evaluations should retain their ordinary corrector: a blanket bypass discards useful corrections along with harmful cache-hit corrections.

Source location and complete-step effect. On four development prompts, disabling the actual OutputRidge hit correctors reduces LPIPS from 0.1051 to 0.0996, whereas disabling misplaced indices yields 0.1162. Disabling only the terminal corrector leaves the result unchanged. A separate same-state diagnostic explains why the predictor path matters: bypassing UniC worsens the immediate post-corrector norm in all 24 cells, yet improves the completed next-step state in 11/12 SeaCache and 7/12 SenCache cells. These cells share four prompt trajectories. The native-scheduler algebra audit independently checks the complete-step identity in all 77 tested grid cells (Appendix E). The same-state diagnostic uses an extra fresh denoiser evaluation solely as a target for measuring error. Its reversal between the immediate and completed states shows that judging a corrector only at its own output can miss the error carried into the next prediction.

Binary gating versus partial correction. A half-strength corrector tests whether approximate evaluations should simply receive a weaker correction. On the eight-prompt controller study, replacing the binary source gate with this rule increases LPIPS from 0.1353 to 0.1386 for SeaCache-0.35, from 0.1259 to 0.1282 for SenCache-$m1$, and from 0.1285 to 0.1307 for SenCache-$m2$. The half-corrector wins only two of eight prompts at each operating point, and all three bootstrap intervals favor the binary gate. These controls support bypassing the current correction on the tested cache hits rather than merely reducing its strength.

Comparison with time-only masks. Frozen-action replay on eight development prompts holds branch decisions constant. SeaCache-0.35 reaches 0.1353 LPIPS with SOURCEMASK, versus 0.1418 with a transferred mask and 0.1372 with a target-selected BestTime mask. For SenCache, BestTime exactly matches SOURCEMASK because its chosen mask aligns with the source trace. This equality is predicted by the fixed-trace result. The masks use development selection information; their construction, intervals, and all replay results are in Appendix E.

5.5 PREDICTOR-MATCHED REFINEMENT

On the held-out predictive split, PW-UNIP improves over SOURCEMASK at both tested solver orders (Table 5). At order 2, the LPIPS reduction is 2.64% with 13/16 prompt wins; the threshold-sweep reanalysis gives a 95% interval [1.23, 4.30]. A frozen K4/K3 output-blending rule improves less. Unlike that control, PW-UNIP retains the K4 hit and adjusts its high-order solver use. At order 3, ordinary/SOURCEMASK/ORDERCACHE LPIPS follows the same ordering.

Table 5: Predictor refinement on p68–p83, two seeds. Each block shares fitted predictors and the fresh/hit schedule, and compares with its order-matched Full-10 reference. The output-blend control changes the returned hit; ORDERCACHE changes its solver treatment.

UniPC order	Solver treatment	LPIPS ↓	SSIM ↑	PSNR ↑
2	SOURCEMASK	0.0936	0.8772	26.422
2	K4/K3 output blend	0.0928	0.8790	26.512
2	ORDERCACHE	**0.0911**	**0.8825**	**26.849**
3	Ordinary	0.1213	0.8513	24.989
3	SOURCEMASK	0.1098	0.8614	25.670
3	ORDERCACHE	**0.1052**	**0.8683**	**26.110**

At order 3, LPIPS falls from 0.1213 to 0.1098 with SOURCEMASK and to 0.1052 with PW-UNIP. The shared K4 fit and source schedule isolate the two solver controls.

The threshold sweep spans $\tau \in \{0, 0.0025, 0.005, 0.01, 0.02\}$. The preselected 0.005 setting gives the lowest mean LPIPS; larger thresholds improve more prompts but yield smaller mean reductions. Applying K4/K3 disagreement to unrelated direct-reuse or SeaCache hit values instead slightly worsens LPIPS. Together with the 0.40% increment on 944 prompts, these results characterize PW-UNIP as a targeted refinement. Appendix D and Table 7 provide the full controls and sweep.

5.6 QUALITATIVE AND PERCEPTUAL ASSESSMENT

Figure 3 compares frames from SeaCache and SenCache at 50 sampling steps. Each row shows the same scene under Full, ordinary caching, and SOURCEMASK, with aligned crops for inspecting local structure. Separately, the eight measured VBench dimensions on the 944-prompt suite remain close across cache treatments, with mixed changes for ORDERCACHE, including lower dynamic degree. This partial, one-video-per-prompt evaluation complements the trajectory-fidelity results; the dimension-wise scores are in Table 20.

5.7 SAMPLING GRIDS AND CROSS-MODEL COMPATIBILITY

The complete-step tradeoff predicts dependence on the grid and the underlying trajectory. We therefore retain the same 30% branch-hit ratio and predeclared direct-reuse schedules across 10, 20, and 50 Wan steps. The LPIPS gain decreases from 8.50% to 2.89%, then reaches 3.41% with an interval spanning zero (Table 6). PSNR also decreases at 20 and 50 steps. The successful 50-step SeaCache/SenCache combinations use different hit mechanisms and schedules, so step count alone does not determine the outcome. The terminal hit has no downstream predictor effect; nonterminal hit placement and solver coefficients govern the grid trend.

Table 6: Operating-range tests with fixed direct-reuse actions. Wan: 16 prompts, two seeds, 30% hit ratio. HunyuanVideo-1.5: eight prompts, two seeds, 25% hit ratio, a common added FlowUniPC solver. Intervals are prompt-bootstrap relative LPIPS effects.

Model / steps	NFE	Ordinary	SOURCEMASK	Wins	Reduction (95% CI)
Wan / 10	14	0.1568	0.1435	16/16	8.50% [7.32, 9.82]
Wan / 20	28	0.1221	0.1186	12/16	2.89% [1.10, 4.96]
Wan / 50	70	0.1394	0.1328	11/16	3.41% [3.27, 7.18]
Hunyuan / 20	30	0.2544	0.2389	7/8	6.09% [3.63, 9.70]

The second-model test uses HunyuanVideo-1.5 480p T2V (Tencent Hunyuan Foundation Model Team, 2025). Its released Euler sampler has no UniC, so all paired candidates use the same added FlowUniPC on the shifted flow grid, with the first sigma changed from 1 to 0.9999 for finite log-SNR. Under five fixed hits in 20 steps, SOURCEMASK reduces LPIPS by 6.09%, and SSIM/PSNR improve as well. This compatibility test changes both the model and solver setup; it shows the provenance gate also transfers to this second model. Full operating points and grid details are in Appendix F.

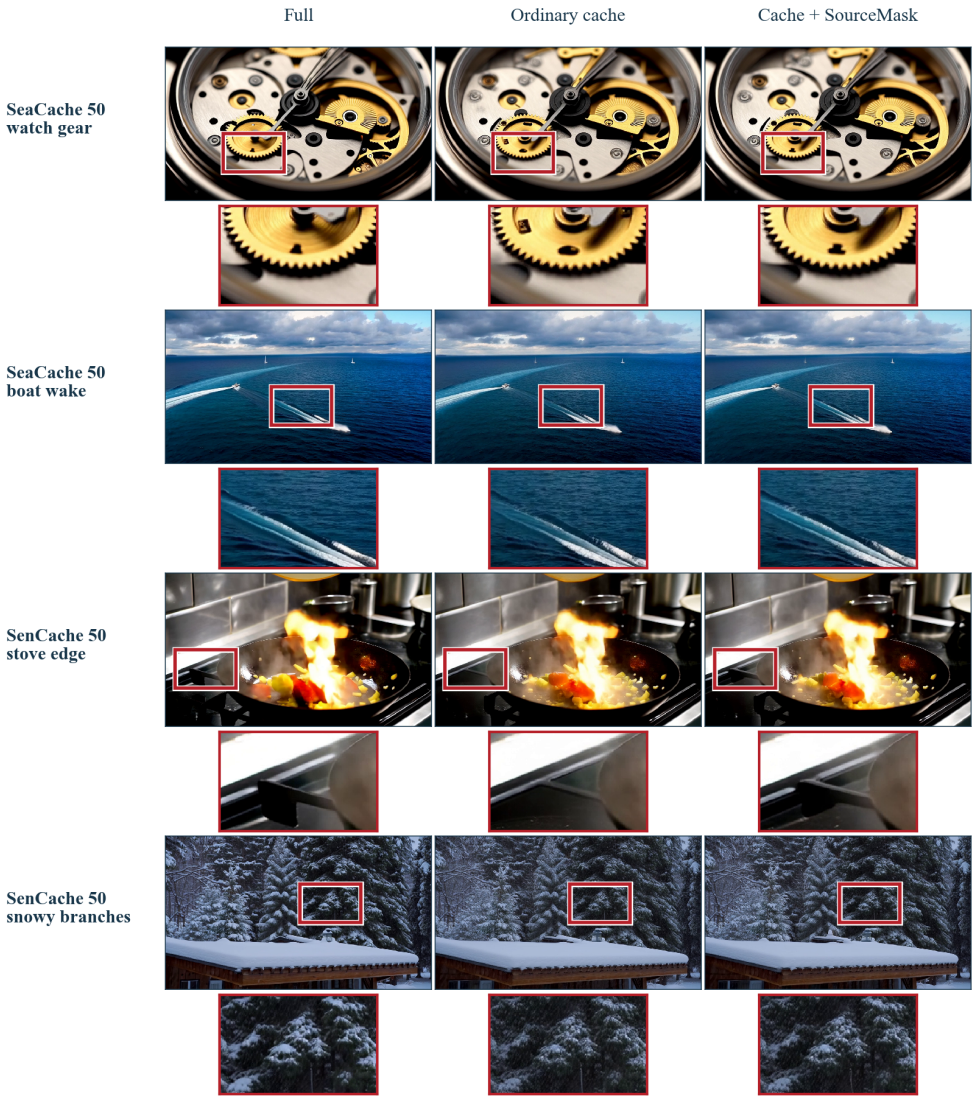


Figure 3: Matched qualitative comparison at 50 sampling steps. The first two rows use SeaCache (watch gears and boat wake); the last two use SenCache (stove edge and snowy branches). Columns show Full, ordinary caching, and caching with SourceMask. Red boxes mark the enlarged regions below each frame.

6 Discussion and Conclusion

Evaluation provenance is a useful part of the cache–solver interface. SourceMask uses this information to control the current correction while retaining approximate evaluations in predictor history. Large-scale and separate Wan experiments demonstrate gains across reuse, prediction, and dynamic controllers at matched evaluation budgets. Branch ablations and complete-step diagnostics explain which decisions matter; the sampling-grid experiments map the operating range, and the HunyuanVideo test shows the gate transfers to a second model. Nested-history agreement provides an additional refinement when it is matched to the active output predictor.

The evidence also points to a clear next step: source labels identify approximate evaluations, while the optimal solver response depends on their error and the solver's local coefficients. Richer source-aware control could retain the simplicity of the interface while adapting the correction strength to this tradeoff. Broader model coverage and a complete perceptual evaluation would test that direction beyond the paired trajectory-fidelity setting studied here.

Ethics Statement

This work studies inference acceleration for pretrained video generators. Reducing computation per generated sample can broaden access to these models and lower the cost of producing misleading or harmful content. Applications should retain the safeguards and usage restrictions of the underlying models. The proposed solver control does not assess the truthfulness or safety of generated videos.

Reproducibility Statement

Sections 3–4 specify the source contract, execution order, and complete-step analysis. Appendices B–D provide the algorithm, full coefficients, and predictor construction; Appendix A gives model settings, prompt/seed splits, metrics, and timing conventions. Extended tables preserve the complete controls and distinguish fixed-action, online, and same-state comparisons.

AI Use Statement

Generative AI tools assisted literature search, methodological discussion, mathematical exposition, interpretation of experimental records, and manuscript drafting and revision. Reported experimental values were taken from the project's experiment records. The authors are responsible for the experiments, claims, references, and final manuscript.

References

Jiazi Bu, Pengyang Ling, Yujie Zhou, Yibin Wang, Yuhang Zang, Dahua Lin, and Jiaqi Wang. DiCache: Let diffusion model determine its own cache. In *International Conference on Learning Representations*, pp. 73778–73803, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/file/78288ef33b18a351c3cd679dc9a15c8d-Paper-Conference.pdf.

Jiwoo Chung, Sangeek Hyun, MinKyu Lee, Byeongju Han, Geonho Cha, Dongyoon Wee, Youngjun Hong, and Jae-Pil Heo. SeaCache: Spectral-evolution-aware cache for accelerating diffusion models. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 14283–14294, 2026. URL https://openaccess.thecvf.com/content/CVPR2026/html/Chung_SeaCache_Spectral-Evolution-Aware_Cache_for_Accelerating_Diffusion_Models_CVPR_2026_paper.html.

Yasaman Haghighi and Alexandre Alahi. SenCache: Accelerating diffusion model inference via sensitivity-aware caching. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 14295–14304, 2026. URL https://openaccess.thecvf.com/content/CVPR2026/html/Haghighi_SenCache_Accelerating_Diffusion_Model_Inference_via_Sensitivity-Aware_Caching_CVPR_2026_paper.html.

Kumara Kahatapitiya, Haozhe Liu, Sen He, Ding Liu, Menglin Jia, Chenyang Zhang, Michael S. Ryoo, and Tian Xie. Adaptive caching for faster video generation with diffusion transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 15240–15252, 2025. URL https://openaccess.thecvf.com/content/ICCV2025/html/Kahatapitiya_Adaptive_Caching_for_Faster_Video_Generation_with_Diffusion_Transformers_ICCV_2025_paper.html.

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 *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 7353–7363, 2025a. URL https://openaccess.thecvf.com/content/CVPR2025/html/Liu_Timestep_Embedding_Tells_Its_Time_to_Cache_for_Video_Diffusion_CVPR_2025_paper.html.

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, pp. 15853–15863, 2025b. URL https://openaccess.thecvf.com/content/ICCV2025/html/Liu_From_Reusing_to_Forecasting_Accelerating_Diffusion_Models_with_TaylorSeers_ICCV_2025_paper.html.

Yuhan Liu, Zongwei Hong, Jinglun Li, Linze Li, Shen Zhang, and Yao Tang. EpaCache: Error-propagation-aware caching for accelerating diffusion-based visual generation. *arXiv preprint arXiv:2608.29264*, 2026. URL https://arxiv.org/abs/2608.29264.

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

Cheng Lu, Yuhao Zhou, Fan Bao, Jianfei Chen, Chongxuan Li, and Jun Zhu. DPM-Solver++: Fast solver for guided sampling of diffusion probabilistic models. *arXiv preprint arXiv:2211.01095*, 2022b. URL https://arxiv.org/abs/2211.01095.

Zhengyao Lv, 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*, pp. 33132–33156, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/file/518046d86bbc41a0707727c38301ad8e-Paper-Conference.pdf.

Xinyin Ma, Gongfan Fang, and Xinchao Wang. DeepCache: Accelerating diffusion models for free. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024.

Zehong Ma, Longhui Wei, Feng Wang, Shiliang Zhang, and Qi Tian. MagCache: Fast video generation with magnitude-aware cache. In *Advances in Neural Information Processing Systems*, 2025. URL https://arxiv.org/abs/2506.09045.

Xurui Peng, Chenqian Yan, Hong Liu, Rui Ma, Fangmin Chen, Xing Wang, Zhihua Wu, Songwei Liu, and Mingbao Lin. ERTACache: Error rectification and timesteps adjustment for efficient diffusion. In *International Conference on Learning Representations*, pp. 82104–82120, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/file/84d395725a9b40cb4a49d84478ac24c7-Paper-Conference.pdf.

Byunggwan Son, Jeimin Jeon, Jeongwoo Choi, and Bumsub Ham. Relational feature caching for accelerating diffusion transformers. In *International Conference on Learning Representations*, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/hash/4019681ef05e8e347c3067d7f973a364-Abstract-Conference.html.

Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=St1giarCHLP.

Tencent Hunyuan Foundation Model Team. HunyuanVideo 1.5 technical report. *arXiv preprint arXiv:2511.18870*, 2025. URL https://arxiv.org/abs/2511.18870.

Vinh Tong, Trung-Dung Hoang, Anji Liu, Guy Van den Broeck, and Mathias Niepert. Learning to discretize denoising diffusion ODEs. In *International Conference on Learning Representations*, pp. 47244–47282, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/file/750046157471c56235a781f2eff6e226-Paper-Conference.pdf.

Wan Team. Wan: Open and advanced large-scale video generative models. *arXiv preprint arXiv:2503.20314*, 2025. URL https://arxiv.org/abs/2503.20314.

Zhou Wang, Alan C. Bovik, Hamid R. Sheikh, and Eero P. Simoncelli. Image quality assessment: From error visibility to structural similarity. *IEEE Transactions on Image Processing*, 13(4): 600–612, 2004. doi: 10.1109/TIP.2003.819861.

Felix Wimbauer, Bichen Wu, Edgar Schoenfeld, Xiaoliang Dai, Ji Hou, Zijian He, Artsiom Sanakoyeu, Peizhao Zhang, Sam Tsai, Jonas Kohler, Christian Rupprecht, Daniel Cremers, Peter Vajda, and Jialiang Wang. Cache me if you can: Accelerating diffusion models through block caching. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024.

Richard Zhang, Phillip Isola, Alexei A. Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 586–595, 2018. URL https://openaccess.thecvf.com/content_cvpr_2018/html/Zhang_The_Unreasonable_Effectiveness_CVPR_2018_paper.html.

Wenliang Zhao, Lujia Bai, Yongming Rao, Jie Zhou, and Jiwen Lu. UniPC: A unified predictor-corrector framework for fast sampling of diffusion models. In *Advances in Neural Information Processing Systems*, volume 36, pp. 49842–49869, 2023. doi: 10.52202/075280-2170. URL https://proceedings.neurips.cc/paper_files/paper/2023/file/9c2aa1e456ea543997f6927295196381-Paper-Conference.pdf.

Wenliang Zhao, Haolin Wang, Jie Zhou, and Jiwen Lu. DC-Solver: Improving predictor-corrector diffusion sampler via dynamic compensation. In *Computer Vision–ECCV 2024*, volume 15069 of *Lecture Notes in Computer Science*, pp. 450–466, 2024. doi: 10.1007/978-3-031-73247-8_26. URL https://www.ecva.net/papers/eccv_2024/papers_ECCV/html/1795_ECCV_2024_paper.php.

Xuanlei Zhao, Xiaolong Jin, Kai Wang, and Yang You. Real-time video generation with pyramid attention broadcast. In *International Conference on Learning Representations*, pp. 3296–3319, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/file/092c2d45005ea2db40fc24c470663416-Paper-Conference.pdf.

Shikang Zheng, Liang Feng, Xinyu Wang, Qinming Zhou, Peiliang Cai, Chang Zou, Jiacheng Liu, Yuqi Lin, Junjie Chen, Yue Ma, and Linfeng Zhang. Forecast then calibrate: Feature caching as ODE for efficient diffusion transformers. *arXiv preprint arXiv:2508.16211*, 2025. URL https://arxiv.org/abs/2508.16211.

Shikang Zheng, Guantao Chen, Qinming Zhou, Yuqi Lin, Lixuan He, Chang Zou, Peiliang Cai, Jiacheng Liu, and Linfeng Zhang. Let features decide their own solvers: Hybrid feature caching for diffusion transformers. In *International Conference on Learning Representations*, pp. 62767–62790, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/file/666dd0d92a64396e753c691db93493d4-Paper-Conference.pdf.

A EXPERIMENTAL PROTOCOL

Appendix roadmap. Appendix A specifies the protocol and accounting. Appendices B–D give the implementation, complete-step derivation, and predictive extension. Appendix E reports mechanism controls, and Appendix F collects full evaluation tables. Appendix G compares related mechanisms.

Model, sampler, and cache sources. Unless noted otherwise, completed video experiments use Wan2.1-T2V-1.3B with native UniPC (Wan Team, 2025; Zhao et al., 2023). The primary evaluation uses 10 sampling steps, shift 5, guidance scale 5, 81 frames, and 832×480 resolution. Within each prompt–seed cell, every candidate starts from identical noise and is compared with the corresponding full-evaluation trajectory on the same time grid. DenoiserReuse-mid performs 14 fresh and 6 reused branch evaluations; DenoiserReuse-fast performs 12 fresh and 8 reused evaluations. OutputRidge uses the K4 predictor of Eq. (21) at cache hits and the fixed fresh-step set $\{0, 1, 2, 3, 4, 6, 8\}$, yielding the hit set $\{5, 7, 9\}$. Its branch coefficients are fitted offline on paid-fresh output histories and then frozen. K3 is fitted independently and supplies the reliability signal for PW-UNIP; the output-blend comparator defined in Appendix D instead combines K4 and K3 in its returned value. All weighted predictions and disagreement reductions are computed in FP32 from BF16 history tensors.

Evaluation splits. The primary SOURCEMASK split has 16 prompt clusters (p44–p59) and two seeds, for 32 matched cells; its configuration was frozen before outcomes were observed. The predictive-extension evaluation uses 16 prompt clusters (p68–p83), disjoint from the primary p44–p59 split, with two seeds and four methods, for 128 rows. The K3/K4 fitting and selection prompts are disjoint from this evaluation. The development public-cache dynamic/replay studies and time-mask table use p20–p27, seed 0, and 50 steps; BestTime selection additionally uses p0–p3 as development prompts. The 10-step location control uses p8/p10/p12/p14, while the same-state diagnostic uses p0/p2/p4/p6, all with seed 0. The held-out controller studies, solver-order comparison, and threshold sweep share p68–p83 and seeds 0/1 with the predictive-extension evaluation. They are separate experiments on a shared evaluation split. The HunyuanVideo compatibility study uses p84–p91 and seeds 0/1. Smaller development studies diagnose the solver mechanism.

Metrics and uncertainty. The primary metric is LPIPS to the matched full-compute UniPC trajectory (LPIPS-to-Full), using AlexNet features (Zhang et al., 2018) and computed per decoded RGB frame in $[-1, 1]$ with equal frame weight. For the two-seed split, we first average seeds within each prompt and then weight prompts equally. Ninety-five-percent intervals use prompt-cluster bootstrap resampling. Relative reductions use the ratio of equal-prompt mean LPIPS values; tables reporting absolute differences or geometric ratios identify those statistics explicitly. The original predictive-extension analysis computes each prompt's mean of the two per-seed log baseline/candidate ratios, averages across prompts, and exponentiates. Its geometric-ratio interval uses 100,000 prompt-cluster draws with seed 3201, as implemented in `phase33_merge_analyze.py`. The 10-step reuse result appears in both the original fixed-budget and later sampling-step analyses. Their recorded bootstrap intervals, [7.29, 9.84] and [7.32, 9.82], respectively, are retained with their corresponding analyses; the paired means and 8.50% point effect are unchanged. Fractional NFE entries are means over prompt–seed cells. PSNR uses $10 \log_{10}(4/\mathrm{MSE})$ on the same range. The SSIM implementation (Wang et al., 2004) uses an 11×11 uniform window, and all candidates are evaluated consistently with this implementation. These metrics measure fidelity to the matched full-compute trajectory, not prompt alignment, aesthetics, or human preference.

Compute accounting. SOURCEMASK does not change the 14/6 or 12/8 counts and adds no tensor cache. For DenoiserReuse-mid, recorded warm end-to-end time is 30.584 s with ordinary UniPC and 30.607 s with SOURCEMASK, while Full-10 takes 41.578 s. The Full-10/cache+SOURCEMASK latency ratio is $1.358\times$, corresponding to a 26.39% reduction in recorded time. Caching supplies the acceleration; SOURCEMASK changes the solver treatment at the same evaluation count. Persistent cache storage remains 7.998 MiB for DenoiserReuse and 31.992 MiB for OutputRidge. The timer sums synchronized denoiser and VAE time after text context is ready and excludes cold model/T5 loading, file output, and metric computation. We use these fixed-order recordings for compute accounting rather than fine-grained kernel benchmarking.

B SourceMask Implementation

Algorithm 1 implements the branch-level source contract. The studied scheduler has an otherwise empty corrector-disable set. The index $i-1$ is the native UniPC convention for suppressing the current corrector at step i. For a scheduler with an existing disable policy, that policy must be composed explicitly with the source gate.

Algorithm 1 SourceMask for branch-specific caching and native UniPC.

Require: initial latent x_0, timesteps $\{t_i\}_{i=0}^{N-1}$, guidance scale w
reset conditional and unconditional cache states
initialize a fresh native UniPC scheduler and its history
for $i = 0, \dots, N-1$ **do**
  $(\widehat{v}_{i,c}, s_{i,c}) \leftarrow \mathrm{CacheOrFresh}_c(x_i, t_i)$
  $(\widehat{v}_{i,u}, s_{i,u}) \leftarrow \mathrm{CacheOrFresh}_u(x_i, t_i)$
  $a_i \leftarrow (s_{i,c} = \mathsf{approx}) \vee (s_{i,u} = \mathsf{approx})$ ▷ $a_i = \neg q_i$ of Eq. (4)
  if $i > 0 \ \wedge \ a_i$ **then**
    `scheduler.disable_corrector`$\leftarrow \{i-1\}$
  else
    `scheduler.disable_corrector`$\leftarrow \varnothing$
  end if
  $\widehat{v}_i \leftarrow \widehat{v}_{i,u} + w(\widehat{v}_{i,c} - \widehat{v}_{i,u})$
  $x_{i+1} \leftarrow \mathrm{UniPCStep}(\widehat{v}_i, t_i, x_i)$
end for
return x_N

C Complete-Step Derivation

Fix the incoming latent, native history, time grid, and realized solver orders. The following derivation tracks the current raw-flow evaluation through scheduler conversion, current correction, history insertion, and prediction. Write $\lambda_i = \log \alpha_i - \log \sigma_i$ for the half-log signal-to-noise ratio. The ρ^C and ρ^P coefficients are the native UniPC `bh2` corrector and predictor coefficients at the realized grid and effective orders (Zhao et al., 2023).

Current-corrector path. For a current corrector at $1 \le i \le N-1$, define $h_i^C = \lambda_i - \lambda_{i-1}$, $\phi_i^C = \mathrm{expm1}(-h_i^C)$, and $B_i^C = \phi_i^C$ for `bh2`. Collecting old-history terms into Q_i gives

$$x_i^C(v) = Q_i - \alpha_i B_i^C \rho_{i,\mathrm{last}}^C \big(m_i(v) - m_{i-1}\big), \tag{13}$$

$$x_i^C(v+e) - x_i^C(v) = k_i^C e, \qquad k_i^C = \alpha_i B_i^C \rho_{i,\mathrm{last}}^C \sigma_i. \tag{14}$$

The sign of B_i^C is retained, and $\rho_{i,\mathrm{last}}^C$ is the coefficient at the realized effective order.

Predictor-history path. For a nonterminal predictor, let its realized effective order be K_i^P, let $h_i^P = \lambda_{i+1} - \lambda_i$, $\phi_i^P = \mathrm{expm1}(-h_i^P)$, $B_i^P = \phi_i^P$, and, for $j = 1, \dots, K_i^P - 1$, define

$$r_j = \frac{\lambda_{i-j} - \lambda_i}{h_i^P}, \qquad D_j = \frac{m_{i-j} - m_i}{r_j}. \tag{15}$$

The update is

$$x_{i+1} = \frac{\sigma_{i+1}}{\sigma_i} x_i^{\mathrm{used}} - \alpha_{i+1}\phi_i^P m_i - \alpha_{i+1} B_i^P \sum_{j=1}^{K_i^P - 1} \rho_j^P D_j. \tag{16}$$

Complete-step coefficients. With

$$A_i = \frac{\sigma_{i+1}}{\sigma_i}, \qquad \beta_i = -\alpha_{i+1}\phi_i^P + \alpha_{i+1} B_i^P \sum_{j=1}^{K_i^P - 1} \frac{\rho_j^P}{r_j}, \tag{17}$$

the current-output coefficients in Eq. (8) are

$$b_{\text{off}} = -\sigma_i \beta_i, \qquad b_{\text{on}} = b_{\text{off}} + A_i k_i^C. \tag{18}$$

The sums are empty when $K_i^P = 1$; at the usual realized second order, only $j = 1$ appears. The plus sign in Eq. (17) follows because m_i appears with a negative sign inside $D_j = (m_{i-j} - m_i)/r_j$. Moreover,

$$d_i = A_i \big[x_i - x_i^C(v_i)\big]. \tag{19}$$

Substitution of $v_i + e_i$ into the two affine maps yields Eq. (9). When $|b_{\text{on}}| > |b_{\text{off}}|$, a direction-independent sufficient condition for a local gain is

$$\|e_i\|_2 > \frac{\|d_i\|_2}{|b_{\text{on}}| - |b_{\text{off}}|}. \tag{20}$$

This sufficient condition depends on the unobserved fresh-output error and is used only to interpret the local decomposition.

Terminal step. With $\sigma_N = 0$ and a lower-order final update, the last predictor is first order and $A_{N-1} = 0$. Its output depends on m_{N-1}, computed before current correction. Hence $T_{\text{on}}(v) = T_{\text{off}}(v)$ at $i = N - 1$. This boundary need not hold for nonzero terminal noise, another scheduler, or a method that re-evaluates after correction.

Fixed-trace equivalence. Given a fixed source set S, SOURCEMASK and the mask in Eq. (11) take the same solver action at every index. Assume their state, history, and cache state agree before step i. They then obtain the same evaluation, apply the same correction decision, insert the same m_i, and produce the same next state and cache state. The base case holds at initialization, so induction proves full-trajectory identity.

D PREDICTIVE-CACHE EXTENSION DETAILS

Exact K4/K3 construction and UniP update. For branch b, let $(v_{s_1,b}, \ldots, v_{s_4,b})$ be the four most recent paid-fresh raw flow outputs. K4 and its independently fitted nested K3 companion are

$$p_{4,i}^b = \sum_{j=1}^{4} a_{4,i,b,j} v_{s_j,b}, \tag{21}$$

$$p_{3,i}^b = \sum_{j=2}^{4} a_{3,i,b,j} v_{s_j,b}. \tag{22}$$

K4 produces the hit; K3 assesses its stability. After branch guidance and scheduler conversion, PW-UNIP applies

$$x_{i+1} = \underbrace{\frac{\sigma_{i+1}}{\sigma_i} x_i^{\text{used}} - \alpha_{i+1} \phi_i^P \widetilde{m}_i}_{\text{base update}} \underbrace{- \alpha_{i+1} B_i^P \omega_i \sum_j \rho_j^P D_j}_{\text{high-order residual}}. \tag{23}$$

Here x_i^{used} is the UniC-corrected state only when the current corrector is enabled; otherwise it is x_i. The K4 output, CFG, and base update are not scaled.

K4/K3 output-blend comparator. The comparator labelled `OutputRidge-OrderBlend` uses the same frozen K4/K3 fits and hit schedule, but modifies the returned branch output. For each branch it computes

$$\eta_{i,b}^{\text{blend}} = \frac{\|p_{4,i}^b - p_{3,i}^b\|_F}{\|p_{4,i}^b\|_F + \varepsilon}, \qquad \beta_{i,b} = \text{clip}(\eta_{i,b}^{\text{blend}}/0.005, 0, 1),$$

$$\widehat{v}_{i,b} = (1 - \beta_{i,b}) p_{4,i}^b + \beta_{i,b} p_{3,i}^b, \tag{24}$$

where ε is the code's numerical norm stabilizer. The threshold and blending rule are frozen; the weight varies with the candidate's own history. The comparator retains the SOURCEMASK corrector gate and the native UniP update. Thus the 0.0928 LPIPS row evaluates output blending rather than constant predictor-residual damping. This definition follows `phase31_erc.py`, lines 248–257 and 354–369, archived in the PW-UniP implementation audit package with source SHA256 `74ee38c03a41c1bd3c7a06d4f2568016ac712727c1853bb60fe3aa57511d33ef`.

Paid-fresh predictor histories. The fixed 10-step predictive-cache schedule refreshes at $\{0, 1, 2, 3, 4, 6, 8\}$ and predicts at $\{5, 7, 9\}$. The corresponding K4/K3 histories are $(1, 2, 3, 4)/(2, 3, 4)$ at step 5, $(2, 3, 4, 6)/(3, 4, 6)$ at step 7, and $(3, 4, 6, 8)/(4, 6, 8)$ at step 9. Conditional and unconditional raw flow outputs are stored separately in BF16 deques of maximum length four; only paid-fresh evaluations enter these deques. Coefficient-weighted predictions, branch-product differences, and reductions are evaluated in FP32. Native UniPC history is separate: it stores converted $m_i = x_i - \sigma_i v_i$ values from both fresh and cache-produced evaluations.

Execution order. At a cache hit, the controller constructs K4 and K3 for both guidance branches, computes η_i and ω_i, marks the K4 evaluation as approximate for SOURCEMASK, applies CFG to K4, and calls the native scheduler. The scheduler first converts the guided raw flow, conditionally applies UniC according to SOURCEMASK, inserts the converted output into native history, and then applies UniP with the high-order residual multiplied by ω_i. At fresh steps, the source gate permits the native correction and $\omega_i = 1$. At terminal step 9, the K4 output and SOURCEMASK remain active, but the effective first-order UniP has no residual for ω_i to scale.

Matched-signal control. We also evaluated K4/K3 disagreement as an auxiliary signal for unrelated hit mechanisms. Adding it to DenoiserReuse changes LPIPS from 0.1177 to 0.1187, with denoiser times of 25.3972 and 25.3980 s. For SeaCache output reuse, LPIPS changes from 0.1233 to 0.1237, again without a speed gain (25.4794 versus 25.4858 s). The direction is unfavorable across the evaluated cells, and dynamic SeaCache traces frequently lack the frozen history patterns needed by K4/K3. Accordingly, PW-UNIP is applied when K4 produces the hit value assessed by its nested companion.

PW-UniP threshold sensitivity. We sweep τ with frozen K4/K3 fitting rules and coefficients, the same source-action schedule, and the same SOURCEMASK gate. Each candidate evolves its own latent and paid-fresh and native histories. Besides quality and latency, the experiment records the mean damping weight $\bar{\omega}$ and the partial/full damping fractions r_p and r_0.

Table 7: Sensitivity of PW-UNIP to the disagreement threshold τ on the held-out predictive split. The $\tau = 0$ row uses the $\tau \to 0^+$ limit; $\bar{\omega}$ is the residual multiplier and r_0 counts fully damped hits. The statistics include the terminal hit, where first-order UniP has no high-order residual to multiply. Time is mean synchronized denoiser time in seconds. Wins compare LPIPS against the paired frozen SOURCEMASK-only rows from the held-out predictive split.

τ	LPIPS	SSIM	PSNR	$\bar{\omega}$	r_p	r_0	Time (s)	Wins
0	0.0930	0.8819	26.871	0.000	0.000	1.000	25.34	9/16
0.0025	0.0922	0.8823	26.871	0.083	0.385	0.615	25.35	11/16
0.005	**0.0911**	**0.8825**	26.849	0.348	0.698	0.302	25.36	13/16
0.010	0.0916	0.8811	26.698	0.622	1.000	0.000	25.35	14/16
0.020	0.0924	0.8795	26.580	0.811	1.000	0.000	25.39	15/16

The prespecified $\tau = 0.005$ has the lowest mean LPIPS and highest mean SSIM among the five settings. Against the same paired SOURCEMASK-only baseline, it reduces mean LPIPS by 2.64% (95% prompt-bootstrap CI $[1.23, 4.30]$; 13/16 prompt wins) without changing the six cached branch evaluations per video. Larger thresholds yield more prompt-level wins (14/16 and 15/16) but smaller average LPIPS reductions (2.18% and 1.31%), illustrating the difference between mean effect and win count. The measured denoiser time varies by less than 0.06 seconds across settings.

E ADDITIONAL CONTROLS AND DIAGNOSTICS

These controls test where correction is suppressed, how branch sources are aggregated, and whether the complete predictor–corrector step explains the observed effect.

E.1 TIME-MASK AND LOCATION CONTROLS

Table 8 reports time-only replay controls, and Table 9 varies the disabled corrector locations.

Table 8: Strong time-only controls under replay on eight development prompts. Values are LPIPS-to-Full; W/T/L compares SOURCEMASK with the time-only control.

Cache point	Time-only control	Source LPIPS	Time LPIPS	Reduction (95% CI)	W/T/L
SeaCache $\delta = 0.35$	Transferred	0.1353	0.1418	4.612% $[1.578, 7.775]$	7/0/1
SeaCache $\delta = 0.35$	Target BestTime	0.1353	0.1372	1.396% $[0.409, 2.669]$	5/3/0
SenCache main$= 1$	Transferred	0.1259	0.1283	1.869% $[-1.217, 4.671]$	6/0/2
SenCache main$= 2$	Transferred	0.1285	0.1308	1.814% $[-1.230, 4.610]$	6/0/2
SenCache main$= 1$	Target BestTime	0.1259	0.1259	0.000% $[0, 0]$	0/8/0
SenCache main$= 2$	Target BestTime	0.1285	0.1285	0.000% $[0, 0]$	0/8/0

The transferred mask begins from a donor modal trace and adjusts the number of suppressed correctors to the target budget; that target suppression count itself was obtained from old target traces. The target BestTime family uses development LPIPS. These controls use development information: the donor came from earlier exposed SeaCache-0.20 results, BestTime LPIPS selection used p0–p3, and the target candidate trace templates were informed by already exposed p20–p27 traces. An internal SeaCache label, `target_trace_p20`, was incorrect: the stored tuple actually matched historical p22/p25/p27 traces. Only the label was corrected; the tuple, selection, outputs, and hashes were unchanged.

Table 9: OutputRidge location controls on four development prompts. Values are LPIPS-to-Full; only the evaluation steps at which UniC is bypassed change (zero-based step labels, not native disable indices).

Solver policy	Evaluation steps with UniC bypassed	LPIPS ↓
Ordinary UniPC	none	0.1051
SOURCEMASK	t5/t7/t9 (actual sources)	**0.0996**
LastOnly	t9	0.1051
Misplaced	t4/t6/t9	0.1162

Partial suppression. For the half-corrector ablation, LPIPS changes from 0.1353 to 0.1386 for SeaCache-0.35, from 0.1259 to 0.1282 for SenCache-main1, and from 0.1285 to 0.1307 for SenCache-main2. The half-corrector wins only 2/8 prompts in each case, and all three bootstrap intervals favor the binary rule.

E.2 IMPLEMENTATION AND COMPLETE-STEP DIAGNOSTICS

The explicit affine coefficients were checked against the unmodified native scheduler on 10-, 20-, and 50-step shift-5 grids, covering 77 non-initial index cells. All 77 CPU/FP32 synthetic fixed-state/history cells pass every predeclared check; maximum normalized displacement discrepancy is 9.677×10^{-8} and maximum normalized squared-identity error is 1.739×10^{-9}. This CPU audit validates the implementation-to-equation mapping; it does not invoke a model, GPU, prompt, or quality outcome.

Complete-step diagnostic. For this diagnostic, one additional denoiser call provides a same-state fresh target. Across 24 such cells, turning UniC off worsens the immediate post-corrector norm in all 24. After completing the next predictor update, however, SOURCEMASK is better in 11/12 SeaCache cells and 7/12 corrected-SenCache cells; the corrected-SenCache middle bucket is negative. The

reversal supports the two-path analysis and shows why a post-corrector norm alone cannot determine the policy. These cells share four prompt trajectories and are not independent samples.

E.3 BRANCH-PROVENANCE GATE

Branch gate ablation. The default gate is the conservative AND rule: UniC is enabled only when both CFG branches are fresh. We compare it with one-branch gates, an OR gate, always-enable, and always-disable with the same cache construction, source-action schedule, and denoiser counts. Each solver candidate retains its own trajectory and histories. A predeclared asymmetric branch-hit schedule on p44–p59 makes these rules distinguishable at the same 15-of-20 branch-evaluation budget. The AND rule reduces LPIPS by 4.66% versus always-enable ordinary caching (95% prompt-bootstrap CI $[3.38, 6.24]$; 16/16 prompt wins), but the conditional-only gate is slightly stronger here at 4.90% reduction (CI $[3.69, 6.37]$; 16/16 wins). Directly comparing these two gates, conditional-only has 0.24% lower LPIPS (CI $[0.07, 0.43]$; 12/16 wins). They differ only at step 5, where the unconditional branch alone is reused. The AND rule defines a conservative provenance contract; this asymmetric schedule identifies a small advantage for the conditional-only variant. OR and unconditional-only are slightly worse than always-enable in LPIPS, and always-disable is less consistent across prompts.

Table 10: Ablation of the branch-provenance gate on the controlled asymmetric CFG-hit schedule, p44–p59, two seeds; all cached rows use 15 branch NFEs. LPIPS, SSIM, and PSNR compare with the paired Full-10 video. Reduction and prompt wins compare each gate with the always-enable ordinary cache.

Gate	Definition	LPIPS	SSIM	PSNR	Reduction (%)	Wins
Always enable	$q_i = 1$	0.5392	0.4016	13.975	0.00	–
Conditional only	$q_i = q_i^c$	**0.5128**	0.4386	15.046	**4.90**	16/16
Unconditional only	$q_i = q_i^u$	0.5409	0.3978	13.914	-0.32	3/16
OR gate	$q_i = q_i^{\mathrm{OR}}$	0.5411	0.3972	13.868	-0.35	1/16
AND gate (SOURCEMASK)	$q_i = q_i^{\mathrm{AND}}$	0.5140	0.4381	15.070	4.66	16/16
Always disable	$q_i = 0$	0.5278	**0.4599**	**16.099**	2.11	9/16

F ADDITIONAL EVALUATION RESULTS

This section reports full metrics for the main comparisons and additional operating points. Unless specified otherwise, experiments use Wan2.1-T2V-1.3B at 832×480, 81 frames, and CFG 5. Each fidelity comparison uses the full-compute trajectory on its own sampling grid.

F.1 DIRECT-REUSE BUDGET SWEEP

The sweep spans f4–f9, where fk denotes k paid-fresh steps on the 10-step grid. At each budget, the ordinary and SOURCEMASK candidates share the source-action schedule. The f9 schedule reuses only at the terminal step and is an exact zero-effect control for the current-corrector gate.

Table 11: Speed–quality sweep for DenoiserReuse on 16 prompts. The ordinary and SOURCEMASK candidates use the same cache actions at each budget; f9 is a terminal zero-effect control.

Budget	Speedup	Ordinary	SOURCEMASK	Difference (95% CI)	Rel.	Wins
f4	2.11×	0.4476	0.4369	0.0108 $[0.0087, 0.0127]$	2.41%	16/16
f5	1.78×	0.3240	0.3055	0.0184 $[0.0139, 0.0231]$	5.69%	15/16
f6	1.54×	0.2344	0.2131	0.0213 $[0.0151, 0.0283]$	9.10%	16/16
f7	1.36×	0.1364	0.1237	0.0126 $[0.0090, 0.0164]$	9.26%	16/16
f8	1.21×	0.0931	0.0823	0.0108 $[0.0076, 0.0144]$	11.62%	15/16
f9	1.10×	0.0436	0.0436	0.0000 $[0.00, 0.00]$	0.00%	0/16

F.2 PREDICTIVE EXTENSION AND SOLVER ORDER

The p68–p83 predictive split uses 16 prompts and two seeds. Each candidate uses the same frozen K4 coefficients, hit schedule, and denoiser budget, with its own states and histories. Table 12 reports the second-order results; Table 13 repeats the comparison with UniPC order 3. ORDERCACHE improves over SOURCEMASK at both orders; the order-3 study additionally compares ordinary caching and preserves the ordinary/SOURCEMASK/ORDERCACHE LPIPS ordering. The threshold study appears in Appendix D.

Table 12: Predictor-matched extension on 16 prompts and two seeds. All variants use the same frozen K4 coefficients, source-action schedule, and denoiser budget; LPIPS is measured against Full-10.

Solver treatment	LPIPS ↓	SSIM ↑	PSNR ↑	Median denoiser time
SOURCEMASK	0.0936	0.8772	26.422	25.324 s
K4/K3 output blend	0.0928	0.8790	26.512	25.344 s
ORDERCACHE	**0.0911**	**0.8825**	**26.849**	25.352 s

Table 13: UniPC order-3 generalization on the p68–p83 split (16 prompts, two seeds). LPIPS/SSIM/PSNR are measured against Full-10.

Solver treatment	LPIPS ↓	SSIM ↑	PSNR ↑	Denoiser time
OutputRidge order-3 (ordinary)	0.1213	0.8513	24.989	25.439 s
+ SOURCEMASK	0.1098	0.8614	25.670	25.499 s
+ PW-UNIP (ORDERCACHE)	**0.1052**	**0.8683**	**26.110**	25.490 s

F.3 HELD-OUT PUBLIC-CONTROLLER COMPOSITION

SeaCache. On p68–p83 with two seeds, SOURCEMASK reduces LPIPS-to-Full under dynamic SeaCache by 16.90% at $\delta = 0.20$ (95% prompt-bootstrap CI $[11.88, 22.69]$, 16/16 prompt wins) and 3.45% at $\delta = 0.35$ (CI $[1.22, 5.41]$, 12/16 wins). Every paired prompt–seed cell has the same branch NFE. Hit locations differ in two of 32 cells at $\delta = 0.20$ and in none at $\delta = 0.35$. These are matched-budget dynamic comparisons; fixed-action replay additionally holds hit locations fixed.

Table 14: Held-out dynamic SeaCache composition on 16 prompts and two seeds. NFE counts fresh CFG-branch evaluations out of 100 possible evaluations; cache ratio and synchronized denoiser time are realized per candidate. Wins count prompt-level LPIPS improvements for SOURCEMASK.

Operating point	Method	LPIPS ↓	SSIM ↑	PSNR ↑	NFE ↓	Cache ratio ↑	Time (s)	Wins
SeaCache $\delta = 0.20$	Ordinary	0.0823	0.8955	26.203	47.94	0.5206	105.6	–
SeaCache $\delta = 0.20$	SOURCEMASK	**0.0684**	**0.9125**	**27.068**	47.94	0.5206	107.4	16/16
SeaCache $\delta = 0.35$	Ordinary	0.1814	0.7745	21.608	34.00	0.6600	75.1	–
SeaCache $\delta = 0.35$	SOURCEMASK	**0.1752**	**0.7803**	**21.681**	34.00	0.6600	78.9	12/16

SenCache. Table 15 reports the 50-step SenCache study on 16 prompts and two seeds at two frozen operating points (start $= 0.045$, main $\in \{1, 2\}$, $K = 3$). Both points improve with a geometric LPIPS ratio near 1.032 ($15/16$ prompts, exact sign $p = 2.6 \times 10^{-4}$) at matched transformer calls (median denoiser latency within ≈ 0.3 s).

Table 16 gives the full metrics for the official-add adapter at two local model-adapted settings. All 32 paired prompt–seed cells at each operating point retain the same realized branch NFE and source trace. The settings differ from the released defaults.

The official-add adapter's recorded dtype paths are shown in Table 17. All 12,800 cached-candidate branch calls in the held-out rerun have BF16 input hidden states and FP32 residuals and post-update

Table 15: SenCache 50-step, enlarged to 16 prompts and two seeds. Values are LPIPS-to-Full; ratio is the geometric ordinary/SOURCEMASK ratio with a bootstrap 95% CI (> 1 favors SOURCEMASK).

Operating point	Ordinary	SOURCEMASK	Geo. ratio (95% CI)	Wins
SenCache main= 1	0.1759	0.1703	1.0322 [1.0194, 1.0460]	15/16
SenCache main= 2	0.1769	0.1713	1.0320 [1.0194, 1.0458]	15/16

Table 16: SenCache composition with the official-add adapter at two local model-adapted operating points (sensitivity 0.045, K3, momentum 1 or 2). The rows do not represent released defaults.

Protocol	Method	LPIPS	SSIM	PSNR	NFE	Cache ratio	Time (s)	Wins	95% CI for R_{LPIPS}
Adapter $m = 1$	Ordinary	0.1759	0.7792	21.474	36.81	0.6319	67.0	–	–
Adapter $m = 1$	SOURCEMASK	**0.1703**	**0.7820**	**21.499**	36.81	0.6319	67.0	15/16	[1.98, 4.37]
Adapter $m = 2$	Ordinary	0.1769	0.7782	21.468	36.00	0.6400	65.5	–	–
Adapter $m = 2$	SOURCEMASK	**0.1713**	**0.7809**	**21.488**	36.00	0.6400	65.5	15/16	[1.99, 4.35]

hidden states. The cache-hit update uses out-of-place addition, preserving the FP32 residual and post-update hidden state.

Table 17: Observed hidden-state and residual dtypes across the held-out SenCache official-add adapter rerun. Counts include both local operating points and both solver treatments, excluding Full-50 rows.

Path	Branch calls	Input hidden	Residual	Output hidden
Fresh	4,660	BF16	FP32	FP32
Cache hit	8,140	BF16	FP32	FP32

F.4 SECOND-MODEL COMPATIBILITY

The HunyuanVideo-1.5 480p T2V test uses eight disjoint prompts (p84–p91), two seeds, paired initial noise, and fixed output reuse at five of 20 denoising nodes (Tencent Hunyuan Foundation Model Team, 2025). The released model uses Euler, which has no UniC corrector. We therefore compare all three trajectories with a common added FlowUniPC on the model's shifted flow grid. The first sigma is changed from 1 to 0.9999 to obtain finite UniPC log-SNR; the remaining time nodes match the released grid.

Table 18: Fixed-budget SourceMask evaluation on HunyuanVideo-1.5 480p T2V with a common added FlowUniPC solver, 20 steps, eight prompts and two seeds. Time is mean synchronized denoiser time per video. All cached rows have the same five hit nodes.

Method	LPIPS ↓	SSIM ↑	PSNR ↑	NFE ↓	Cache ratio ↑	Time (s)	Reduction (%)	Wins
Ordinary direct reuse	0.2544	0.6960	20.265	30	0.25	119.6	–	–
SOURCEMASK	0.2389	0.7104	20.597	30	0.25	117.5	6.09	7/8

SOURCEMASK reduces LPIPS by 6.09% relative to ordinary reuse (95% prompt-bootstrap CI for reduction $[3.63, 9.70]$; 7/8 prompt wins). SSIM and PSNR also improve, at the same 30 branch NFEs and 25% cache ratio. This comparison tests model compatibility under a common adapted solver, and shows the provenance gate transfers to this model and grid.

F.5 SAMPLING-STEP ROBUSTNESS

A frozen direct-reuse experiment compares 10-, 20-, and 50-step grids on p44–p59 with two seeds. For each $N \in \{10, 20, 50\}$, the ordinary and SOURCEMASK candidates use

Table 19: Dynamic combinations on eight development prompts at 50 steps. Values are LPIPS-to-Full. SenCache uses official out-of-place addition semantics. Source traces and realized NFE can differ between solver treatments in these development runs.

Controller	Ordinary	SOURCEMASK	Reduction (95% CI)	Wins
SeaCache $\delta = 0.20$	0.0646	**0.0599**	**7.174%** $[2.501, 15.210]$	8/8
SeaCache $\delta = 0.35$	0.1437	**0.1353**	**5.832%** $[3.559, 8.892]$	8/8
SenCache main$= 1$	0.1290	0.1259	2.362% $[-0.942, 5.307]$	7/8
SenCache main$= 2$	0.1315	0.1285	2.296% $[-0.989, 5.217]$	6/8

the same predeclared source trace, paired initial noise, and a 30% branch-hit ratio. Paid-fresh branch flows are cached in BF16 and returned as FP32 on hits, matching the primary direct-reuse adapter. The respective zero-based hit lists are $\{5, 7, 9\}$, $\{5, 8, 11, 13, 16, 19\}$, and $\{5, 8, 11, 14, 18, 21, 24, 27, 30, 33, 36, 40, 43, 46, 49\}$; no list is selected using the final quality outcome.

For a fixed step count, the primary quantities are

$$\Delta_N = \mathrm{LPIPS}_{\mathrm{ordinary},N} - \mathrm{LPIPS}_{\mathrm{SM},N}, \qquad R_N = 100\left(1 - \frac{\mathrm{LPIPS}_{\mathrm{SM},N}}{\mathrm{LPIPS}_{\mathrm{ordinary},N}}\right)\%. \tag{25}$$

Every list contains the final step, so the native terminal zero-effect condition is present by construction; effects in the table include its inert hit.

F.6 DEVELOPMENT PUBLIC-CACHE COMBINATIONS

We combine SOURCEMASK with SeaCache (Chung et al., 2026) at $\delta \in \{0.20, 0.35\}$ and SenCache (Haghighi & Alahi, 2026) at two locally frozen settings (start $= 0.045$, main $\in \{1, 2\}$, $K = 3$). The SenCache points are locally frozen operating settings rather than paper-table defaults. Every reported SenCache value uses the official out-of-place residual-addition semantics with FP32 promotion; results from an earlier in-place BF16 adapter are excluded.

Table 19 reports dynamic 50-step combinations. SeaCache improves at both settings, with all eight prompts improving and confidence intervals above zero. Corrected SenCache also improves mean LPIPS by 2.36% and 2.30%, with 7/8 and 6/8 prompt wins; its wider intervals cross zero on this eight-prompt study. Because changing the solver can change a dynamic controller's later source trace, realized NFE need not match within these public-cache pairs. This development table measures coupled-system behavior. The held-out controller comparisons above audit equal realized branch NFE in every paired cell.

F.7 PARTIAL VBENCH DIMENSIONS

Table 20 complements LPIPS-to-Full with the eight measured perceptual dimensions. The cache variants remain close on most scores; ORDERCACHE lowers dynamic degree relative to ordinary OutputRidge. We report the dimensions separately because this single-video protocol does not produce the official overall VBench score.

Table 20: Eight officially runnable VBench dimensions on the 944-prompt suite (higher is better). This is a partial VBench: only 8 of 16 official dimensions run under our current dependencies, each prompt generates a single video rather than the official five, and per-dimension prompt counts follow the official assignment. These scores complement, and do not substitute for, an official overall VBench score.

Solver treatment	Aesth.	Style	Bg. cons.	Dyn. deg.	Imaging	Motion	Subject	Temporal
Full-10	0.6131	0.2245	0.9726	0.6389	0.6339	0.9736	0.9381	0.9869
OutputRidge (ord.)	0.6063	0.2235	0.9722	0.6528	0.6246	0.9730	0.9352	0.9864
SOURCEMASK	0.6064	0.2237	0.9723	0.6528	0.6241	0.9730	0.9352	0.9864
ORDERCACHE	0.6062	0.2238	0.9723	0.6389	0.6247	0.9731	0.9351	0.9865

G EXTENDED RELATED WORK

When to refresh. Training-free caches exploit temporal redundancy while deciding when expensive computation must be refreshed. DeepCache reuses high-level U-Net features (Ma et al., 2024); block caching searches module-level reuse patterns (Wimbauer et al., 2024); PAB broadcasts attention outputs over module-dependent ranges (Zhao et al., 2025); and FasterCache additionally shares computation across guidance branches (Lv et al., 2025). Adaptive controllers derive refresh decisions from inexpensive signals: TeaCache uses timestep-modulated inputs (Liu et al., 2025a), AdaCache adapts to sample motion (Kahatapitiya et al., 2025), MagCache models residual magnitude (Ma et al., 2025), SenCache uses latent and timestep sensitivity (Haghighi & Alahi, 2026), and SeaCache filters its proxy according to spectral evolution (Chung et al., 2026). EpaCache calibrates the downstream impact of isolated reuse errors and allocates timestep-specific thresholds accordingly (Liu et al., 2026). These methods determine refresh actions. SOURCEMASK passes their fresh/cache-produced decisions to UniPC.

What to return on a hit. A second line of work instead improves the approximate value itself. TaylorSeer replaces direct reuse with temporal feature forecasting (Liu et al., 2025b); DiCache combines a shallow refresh probe with multi-cache trajectory alignment (Bu et al., 2026); and relational feature caching couples input–output prediction with an error-aware scheduler (Son et al., 2026). FoCa casts hidden-feature evolution as a feature ODE and uses a BDF2 predictor with Heun-style calibration (Zheng et al., 2025), while HyCa assigns ODE-inspired forecast rules to feature dimensions with different dynamics (Zheng et al., 2026). These numerical ideas operate inside cache-value construction. Our K4 predictor likewise constructs a cache-hit value from history; the distinct contribution of PW-UNIP is to derive a matched nested-history signal from that active predictor and expose it to the outer UniP update.

How the sampler consumes evaluations. Diffusion solvers such as DDIM, DPM-Solver, and UniPC reduce sampling error or denoiser evaluations through improved trajectory integration (Song et al., 2021; Lu et al., 2022a;b; Zhao et al., 2023). LD3 learns a denoising-ODE time grid offline (Tong et al., 2025). DC-Solver compensates the mismatch between a post-corrector state and an available pre-correction evaluation by modifying predictor–corrector history (Zhao et al., 2024). ERTACache combines cache-error rectification with cache-dependent timestep adjustment (Peng et al., 2026). SOURCEMASK addresses a different and directly observable interface: it conditions the current UniC path on whether the present evaluation was freshly computed or produced by a cache. It leaves the cached value, native history insertion, and time grid intact, allowing the same policy to compose with reuse, prediction, and dynamic refresh controllers. Table 21 summarizes the signal and intervention of each mechanism.

Table 21: Operational taxonomy of closely related cache and solver mechanisms. Source-conditioned UniC denotes direct control by the fresh/cache-produced provenance of the current evaluation.

Method	Primary signal	Direct intervention	Outer time grid	Source-conditioned UniC
SeaCache / SenCache	spectral change / sensitivity	refresh decision	fixed	no
EpaCache	calibrated downstream impact	refresh thresholds	fixed	no
DiCache	shallow probe, cache trajectory	refresh and hit value	fixed	no
FoCa / HyCa	hidden-feature dynamics	cache-hit feature forecast	fixed	no
DC-Solver	predictor–corrector mismatch	solver-history compensation	fixed	no
ERTACache	residual profile and error model	hit value and interval	adjusted	no
SOURCEMASK	branch execution provenance	current UniC	fixed	**yes**
ORDERCACHE	provenance and K4/K3 agreement	UniC and UniP residual	fixed	**yes**