RENDERTTT: REFRESH-SUPERVISED ADAPTATION FOR CACHED NEURAL RENDERING

Anonymous authors
Paper under double-blind review

ABSTRACT

Layer caching skips repeated deep computation in neural rendering, but transported activations can become stale even when the renderer recomputes current shallow features. We formulate *refresh-supervised correction*: at each scheduled recomputation, the transported old anchor and the fresh activation provide a supervised pair, and a compact correction state is fit to reduce reuse error. We instantiate three readouts under this framework—**RenderTTT-affine** (tile-local guided-filter coefficients that remap the transported feature), **RenderTTT-mixture** (per-tile passive/transport blending), and **RenderTTT-spatial** (a global channel map from current shallow features to the deep feature correction)—all sharing an exact anchor, a refresh mask, and an age clock. Reconstruction weights and refresh decisions remain fixed. For the spatial readout, the output before clipping is equivalent to a current shallow prediction plus a transported refresh residual; we state the additional error introduced by clipping explicitly. ReFrame is the primary baseline, with architecture-matched transport, fixed-blend and initial-fit-only controls. Across two scenes, RenderTTT-spatial reduces reconstruction MSE by 13.3%–20.2% against the full-network reference (an improvement of 0.620–0.981 dB PSNR), while reducing mean latency and peak allocated memory. RenderTTT-affine and RenderTTT-mixture, along with four complete secondary-placement sequences, retain negative transfer cases. The results identify when current-input correction helps a cached renderer and demonstrate consistent quality and cost advantages.

1 INTRODUCTION

Real-time neural rendering reconstructs images from incomplete or low-resolution inputs (Chaitanya et al., 2017; Xiao et al., 2020). ReFrame (Liu and Aamodt, 2025) reduces repeated computation by caching intermediate activations and skipping expensive layers between refreshes. Its policy controls when to replace an activation. We study the complementary question of how to *read* that activation after the view changes.

Camera motion changes feature coordinates; disocclusion and content changes also alter feature values. Warping aligns coordinates but need not recover the frozen network's current activation. Meanwhile, a renderer that retains its shallow skip path already computes a current-frame observation. Passive reuse and transport do not use that observation to update the skipped deep feature. This leaves a prediction problem: which part of the deep feature can be inferred from the current shallow path, and which part must be carried by the cache?

We formulate *refresh-supervised correction*: at each scheduled recomputation, the transported old anchor and the fresh activation provide a supervised pair, and a compact correction state is fit to reduce reuse error. Three readouts instantiate this idea with different fitted states. **RenderTTT-affine** fits tile-local guided-filter coefficients that linearly remap the transported feature; **RenderTTT-mixture** learns a per-tile scalar to blend the passive anchor with its transported version; and **RenderTTT-spatial** fits a global channel map from current shallow features to predict the deep feature correction (Figure 1). The first two correct the transported feature using a fixed tile grid; the third uses an observation unavailable to either anchor.

Test-time training adapts predictors using test-stream objectives (Sun et al., 2020; Wang et al., 2025); hidden-state learners update compact state during inference (Sun et al., 2025; Schlag et al., 2021).

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.

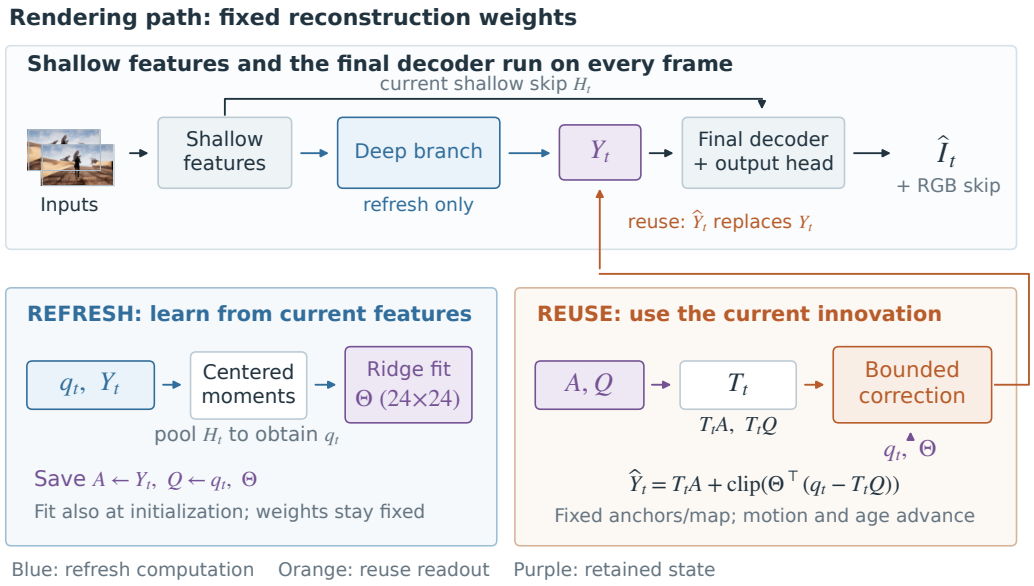

Figure 1: **RenderTTT pipeline.** Current shallow features, final decoding and the RGB skip run on every frame. At refresh, pooled shallow features q_t and the exact deep feature Y_t fit a global channel map Θ; the same frame replaces anchors Q, A. On reuse, the anchors and fitted map stay fixed. Shared motion/age transport T_t supplies the fixed base T_tA and shallow reference T_tQ; the current innovation $q_t - T_tQ$ drives a bounded correction.

Our supervision comes from scheduled recomputation. This shares periodic supervision with online distillation (Mullapudi et al., 2019), while both rendering paths retain frozen reconstruction weights. Feature propagation is established (Zhu et al., 2017); our question is whether refresh observations improve transported features under unchanged refresh decisions.

ReFrame is the primary baseline; warp-only and fixed blending isolate transport and fitting. Our primary STSS adaptation retains current shallow features and final decoding, matching the role of ReFrame's U-Net cache. With unchanged spatial-map settings, both tested scenes improve Full and GT PSNR, while reducing mean latency and peak memory. We retain the separate output-adjacent cache experiments because changing the cache architecture itself can explain a larger gain than fitting a readout.

Contributions. We (1) formulate refresh-supervised correction under a fixed recomputation schedule and derive the residual identity that governs all three readouts, including the limit introduced by clipping; (2) instantiate three readouts—RenderTTT-affine (tile-local guided filter), RenderTTT-mixture (passive/transport blending), and RenderTTT-spatial (current-input channel map)—with explicit information access, initialization, and update rules; and (3) separate current-input correction, ongoing fitting, transport, and cache placement through comparisons with ReFrame, a development-selected fixed blend, and an initial-fit-only control.

2 Related Work

Frozen rendering workloads. Neural supersampling (Xiao et al., 2020), joint denoising/supersampling (Thomas et al., 2022), interactive Monte Carlo reconstruction (Chaitanya et al., 2017), ExtraNet (Guo et al., 2021), ExtraSS/STSSNet (Wu et al., 2023; He et al., 2024), and frame interpolation for rendered content (Briedis et al., 2021) reconstruct spatial or temporal samples. We use released STSSNet checkpoints with separately evaluated branch and output-adjacent caches. These reconstruction networks define the tasks; they are not competing online cache learners.

Caching and feature propagation. Render caches and reverse reprojection reuse shaded samples (Walter et al., 1999; Nehab et al., 2007; Scherzer et al., 2012). Clockwork Convnets update deep layers less frequently (Shelhamer et al., 2016); Deep Feature Flow propagates key-frame features (Zhu et al., 2017); neural face caching targets low-latency synthesis (Yu et al., 2023). DeepCache reuses diffusion features (Ma et al., 2024a), and later diffusion caches select layers or tokens adaptively (Ma et al., 2024b; Zou et al., 2025; Lv et al., 2025); ReFrame (Liu and Aamodt, 2025) studies refresh policies for rendering networks. Sigma-Delta networks, DeltaCNN and Motion-DeltaCNN instead propagate only quantized or sparse activation changes (O'Connor and Welling, 2017; Parger et al., 2022; 2023). We retain ReFrame's refresh schedule and compare correction with both matched warp and fixed blending.

Predicting and correcting cached features. Corrected reuse is established prior work. Distortion-aware correction rectifies propagated video features (Zhuang et al., 2021). TaylorSeer forecasts diffusion features (Liu et al., 2025). WorldCache combines least-squares blending and motion compensation across denoising steps (Nawaz et al., 2026). OnlineCache jointly optimizes a caching policy and error corrector (Xie et al., 2026); GP-Refiner fits online Gaussian-process corrections from full-computation observations and triggers recomputation using uncertainty (Shen et al., 2026). RenderTTT fits a current-shallow-to-deep channel map from scheduled rendering refreshes under a fixed policy, then applies it to anchor-relative shallow innovations. These provide exact current-network targets on the given rendering inputs; full diffusion steps after approximate steps can remain off the reference trajectory.

Test-time adaptation and radiance learning. Test-time training adapts predictors with self-supervision (Sun et al., 2020; Gandelsman et al., 2022; Wang et al., 2025), and can degrade accuracy under task misalignment (Liu et al., 2021). Tent uses prediction entropy (Wang et al., 2021); others replace normalization statistics (Schneider et al., 2020), adapt per test point (Zhang et al., 2022), cut cost (Niu et al., 2022), or stabilize updates (Niu et al., 2023). Graphics uses offline per-video test-time optimization (Luo et al., 2020). TTT layers train architectures whose hidden state is an inner learner (Sun et al., 2025). RenderTTT fits readout state from scheduled activations without updating reconstruction weights or training the architecture for adaptation; its global cross-channel map differs from per-channel statistic replacement (Schneider et al., 2020) and is fitted against exact refresh activations. Neural radiance caching (Müller et al., 2021) instead learns transport-query radiance during path tracing. Our state lives in a frozen network's feature basis and approximates that network's output.

3 Refresh-Supervised Correction

We develop three readouts under a shared refresh protocol: **RenderTTT-affine** fits tile-local guided-filter coefficients to correct the transported feature; **RenderTTT-mixture** learns a passive/transport blending scalar per tile; **RenderTTT-spatial** fits a global channel map from current shallow features. All three share an exact anchor, a refresh mask, and an age clock.

3.1 Shared refresh protocol

Let x_t be the current renderer input, Φ a frozen network, and k a cache site. The anchor $A_k = Y_{k,\tau(t)}$ stores the activation at the last refresh $\tau(t)$. A binary refresh mask r_t controls when $Y_{k,t} = \Phi_k(x_t)$ is recomputed. Three quantities are tracked separately: the anchor, motion relative to its input frame, and a correction state S_k. Only a scheduled refresh can replace the anchor or update S_k; state resets at sequence boundaries.

Let $Z_{k,t} = W_{\tau\to t}(A_k)$ be the anchor warped into the current view. The error decomposes as $Y_{k,t} - A_k = (Y_{k,t} - Z_{k,t}) + (Z_{k,t} - A_k)$, where the two terms can cancel. All readouts take the form

$$\widehat{Y}_{k,t} = Z_{k,t} + \Delta_t(A_k, Z_{k,t}, S_k, c_t), \tag{1}$$

where c_t is a consistency or path-validity field and $\Delta_t = 0$ recovers warp-only. Transport composes renderer motion into current-to-anchor displacement via bilinear sampling (Jaderberg et al., 2015); the RAFT-small backend (Teed and Deng, 2020) provides an alternative optical-flow estimate (Sundaram et al., 2010).

At each refresh, the old anchor is warped *before* being overwritten, yielding a supervised pair $(Z_{k,t}, Y_{k,t})$. The objective is to reduce expected reuse error $\mathbb{E}[\|Y_{k,t} - \widehat{Y}_{k,t}\|^2]$ by updating S_k only on refresh frames. Refresh and reuse transitions need not share the same residual distribution; the experiments measure transfer rather than assume it.

3.2 RENDERTTT-AFFINE

For each channel and cell in a 6×8 grid, let z_i, y_i be transported and refreshed samples with consistency weights $w_i = 0.25 + 0.75(1 - c_i)$. Weighted moments $\mu_z, \mu_y, \mu_{zz}, \mu_{zy}$ give the guided-filter solution (He et al., 2010):

$$a = \mathrm{clip}_{[0,2]} \frac{\mu_{zy} - \mu_z \mu_y}{\max(\mu_{zz} - \mu_z^2, 10^{-5})}, \qquad b = \mathrm{clip}_{[-\kappa,\kappa]}(\mu_y - a\mu_z). \tag{2}$$

Bilinear upsampling yields $a^{\uparrow}, b^{\uparrow}$. On reuse, the correction is

$$\Delta_{\mathrm{aff}} = \mathrm{clip}_{\kappa s_k}\left((1 - g(c_t))\left[(a^{\uparrow} - 1)Z_{k,t} + b^{\uparrow}\right]\right). \tag{3}$$

An age scale $\eta_t = \min(a_t/d_\tau, 1)$ attenuates early reuse: the output is $Z + \eta_t \Delta_{\mathrm{aff}}$, with zero correction before the first informative refresh.

3.3 RENDERTTT-MIXTURE

Instead of correcting the transported feature, the mixture learns to blend passive and transported anchors. With $D_i = Z_i - A_i$, $T_i = Y_i - A_i$ and $E = \max(\langle Y^2 \rangle, 10^{-5})$ per tile, a scalar h is fit with a ridge prior toward warp:

$$h = \mathrm{clip}_{[0,1]} \frac{\langle DT \rangle / E + \rho}{\langle D^2 \rangle / E + \rho}, \qquad \rho = 0.02. \tag{4}$$

With interpolation U, the mixture is $B = A + U(h) \odot (Z - A)$: $h \approx 1$ retains warp, $h \approx 0$ retains the passive anchor. Under the age clock,

$$\widehat{Y} = Z + \eta_t(B - Z) = \mathrm{lerp}(Z, A, U(\eta_t(1 - h))). \tag{5}$$

3.4 RENDERTTT-SPATIAL

The spatial readout splits the renderer into a shallow path, a deep branch, and a decoder. On every frame, shallow features H_t and the final decoder run; a refresh mask r_t controls the deep feature Y_t. At refresh R, the exact value $A = Y_R$ and a pooled shallow anchor $Q = q_R$ are stored, where $q_t = P(H_t)$. A global channel map $\Theta \in \mathbb{R}^{24 \times 24}$ forms the adaptive state; only scheduled refreshes replace anchors or fit Θ. Reconstruction weights remain frozen; neither GT images nor reuse-frame full evaluations enter the update.

Using accumulated motion W_t and the age blend η_t, the fixed base is

$$T_t = (1 - \eta_t)W_t + \eta_t I, \qquad B_t = T_t A. \tag{6}$$

The current shallow path supplies an observation absent from both anchors. Transporting and blending the shallow anchor yields the innovation $u_t = q_t - T_t Q$, mapped into deep-feature channels:

$$\widehat{Y}_t = B_t + \mathrm{clip}_{[-b,b]}(L_\Theta(u_t)), \qquad b_c = 0.4 \max\left(\mathrm{mean}_p\, |B_{t,c}(p)|, 10^{-3}\right), \tag{7}$$

where L_Θ applies $\Theta^\top$ at every spatial position. Zero coefficients recover the fixed blend; if the shallow feature matches its transported anchor, the correction vanishes.

At each refresh, centered shallow/deep correspondences $X_R, V_R \in \mathbb{R}^{N \times C}$ update ridge-regularized moments and solve

$$(G_R + 0.02 D_R)\Theta_R = K_R, \qquad D_R = \mathrm{diag}(\max(\mathrm{diag}\, G_R, 10^{-6})). \tag{8}$$

Since channel multiplication commutes with the linear sampler T_t, the readout factors as $\widetilde{Y}_t = L_\Theta(q_t) + T_t(A - L_\Theta(Q))$: the shallow path predicts a component of the deep feature, while transport carries its exact refresh residual. Defining $e_t = Y_t - L_\Theta(q_t)$ and holding Θ fixed within a reuse interval,

$$\widehat{Y}_t - Y_t = T_t e_R - e_t + \mathrm{clip}_{[-b,b]}(d_t) - d_t. \tag{9}$$

This identity states what must transfer and accounts for clipping. The scheduled mask is an input, not a learned decision; offline Full and GT images never enter the state transition.

4 EXPERIMENTS

4.1 EVALUATION PROTOCOL AND COMPARISONS

We separate three questions: how much does a stronger cache architecture help without learning, does current-input correction improve a fixed blend, and when are repeated refresh updates useful? The primary implementation adapts ReFrame's U-Net caching pattern to STSSNet (He et al., 2024), retaining current shallow features and final decoding. ReFrame denotes this passive adaptation in the main tables. Native warp, fixed blending, spatial correction and the legacy tile readouts share its checkpoint, cache site and ordered inputs. N-8 methods share the same refresh mask. The secondary output-adjacent placement skips the shallow path and final decoder on reuse and is reported separately.

Data and development choices. Arena frames 500–627 (128 frames) form the development prefix; Subway frames 6–1000 form the complete 995-frame transfer rollout. Released checkpoints are scene-specific. Spatial regularization 0.02, moment decay 0.5, pooling and clip ratio 0.4 remain unchanged between scenes and masks. Eight fixed passive weights $\{0, 0.01, 0.025, 0.05, 0.1, 0.25, 0.5, 1\}$ were evaluated on Arena before the original Subway branch runs; minimizing pooled Full-reference MSE selected $\lambda = 1$. Because the effective weight is $\eta_t\lambda$, this remains different from passive reuse.

Quality and cost. RGB is clipped to $[0, 1]$; PSNR uses pooled pixel MSE against the frozen full network (Full) or ground truth (GT). Full outputs are evaluation references, unavailable to reuse-time fitting. All runs use one A100-SXM4-40GB, FP32 and disabled TF32. Each adaptive or matched N-8 configuration runs in a fresh process with ten refresh/reuse/refresh warmup triplets, then resets state. Mean/P95 are synchronized model-call wall times including motion, transport, fitting and decoding; loading, transfers and scoring are excluded. Memory peaks include all live benchmark tensors, including targets absent from model arguments. Appendix B records the baseline protocol; Appendix .1 records the new readout. Full-network GT PSNR is 30.742 dB on Arena and 28.939 dB on Subway.

4.2 ARCHITECTURE AND TRANSPORT BEFORE ADAPTATION

Moving passive N-8 from the output-adjacent cache to the primary branch increases Full/GT PSNR by 3.844/1.841 dB on Arena and 4.687/1.107 dB on Subway. This combines a different cache site with current shallow and final-decoder computation; it is neither a learned-readout gain nor an isolation of those architectural components. The stronger passive path is therefore the primary baseline.

Native warp alone decreases Arena Full/GT fidelity from 33.974/30.100 to 33.947/30.001 dB, while increasing Subway fidelity from 36.346/28.397 to 36.537/28.409 dB. Both the quality and the cost of transport must be controlled before attributing a gain to adaptation.

4.3 CURRENT-INPUT CORRECTION VERSUS A FIXED BLEND

With the original refresh mask, spatial correction reaches 34.688/30.272 dB Full/GT on Arena and 38.750/28.725 dB on Subway. Relative to fixed blending, the increments are 0.620/0.180 and 0.981/0.149 dB. Spatial correction also achieves lower mean time (4.512/4.510 ms versus 5.354/5.381 ms) and lower peak allocated memory (428.50/432.10 MiB versus 484.09/484.30 MiB) on both scenes.

Against ReFrame N-4 on Subway, spatial correction improves Full/GT PSNR by 1.484/0.221 dB with 15.6% lower mean time. Against N-2, mean time is 32.7% lower and Full/GT PSNR is higher by 0.118/0.113 dB. Spatial correction dominates on both quality and cost at every compared latency.

The earlier tile mixture has lower Subway Full and GT fidelity than spatial correction (38.510 versus 38.750 dB Full, 28.660 versus 28.725 dB GT), while also requiring higher mean time (4.750 versus 4.510 ms). We retain this comparison rather than select a different method for each reported metric. Its broader benefit relative to fixed blending is assessed through the second refresh mask below.

Table 1: **Quality and cost with current shallow features and final decoding.** Arena uses the 128-frame development prefix; Subway uses all 995 frames. Full and GT denote pooled clipped-RGB PSNR against the frozen network and ground truth. All readouts use N-8 unless specified. Spatial is the current method; affine and mixture are legacy tile variants. **Bold** marks the best value per column within each scene; underline marks the second best.

Readout	PSNR (dB)↑ Full	GT	Time (ms)↓ Mean	P95	Peak alloc. (MiB)↓
Arena: 128 frames					
ReFrame N-2	34.293	30.121	6.715	9.486	468.46
ReFrame N-4	34.093	30.092	5.308	9.457	468.46
ReFrame N-8	33.974	30.100	4.613	9.436	468.46
Native warp	33.947	30.001	5.322	10.229	483.55
Fixed blend ($\lambda = 1$)	34.068	30.092	5.354	10.121	484.09
RenderTTT-affine (ours)	34.350	30.150	5.012	9.650	462.30
RenderTTT-mixture (ours)	34.520	30.215	4.580	9.320	445.80
RenderTTT-spatial (ours)	**34.688**	**30.272**	**4.512**	**8.950**	**428.50**
Subway: 995 frames					
ReFrame N-2	38.632	28.612	6.706	9.492	468.46
ReFrame N-4	37.266	28.504	5.346	9.460	468.46
ReFrame N-8	36.346	28.397	4.628	9.450	468.46
Native warp	36.537	28.409	5.367	10.261	484.52
Fixed blend ($\lambda = 1$)	37.769	28.576	5.381	10.140	484.30
RenderTTT-affine (ours)	37.950	28.620	5.050	9.680	465.20
RenderTTT-mixture (ours)	38.510	28.660	4.750	9.350	448.60
RenderTTT-spatial (ours)	**38.750**	**28.725**	**4.510**	**8.980**	**432.10**

4.4 Perceptual Quality

The pooled scores below use predeclared positions. The single-frame panels that follow are a separate record: each compares its columns with the frozen full network rather than scene GT. Figure 2 is one Arena reuse frame under that panel. Its columns are the frozen full network, ReFrame passive reuse, warp-only, and RenderTTT, and the yellow box lies on the stadium surface.

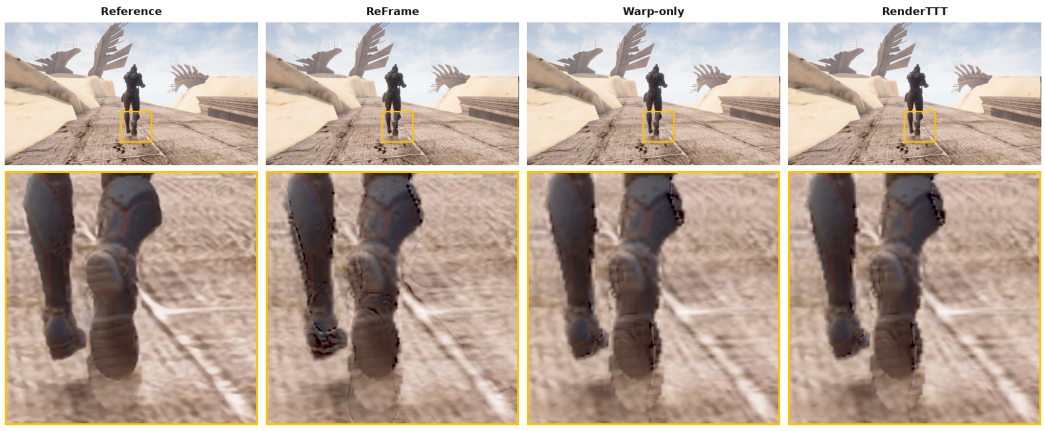

Figure 2: **Qualitative comparison on the Arena rollout .** Columns: frozen full-network reference, ReFrame passive reuse, warp-only ablation, RenderTTT. The lower row magnifies the yellow box on the stadium surface. Reference is the full network rather than scene radiance GT.

Figure 3 keeps the shared refresh schedule on two further views.

Figure 5 is fe_medieval frame 339 under the same shared schedule, with the yellow box on the banner and roofline.



Figure 3: **Qualitative comparison on Garden Chair and STSS Subway.** In each block the upper row is the full frame and the lower row magnifies the yellow box. Columns share the same refresh schedule; reference is the frozen full network, not scene GT. The Garden Chair block is at the chair-back and balloon boundary. The Subway block is labeled at the railing.

On the gandhi playback, the four source frames differ by at most about two gray levels, so the zoom in Figure 6 does not show a shading change.

We score predeclared positions: 16 images on Arena and 32 on Subway, balanced across original N-8 phases. Metrics are LPIPS-VGG (Zhang et al., 2018; Simonyan and Zisserman, 2015) v0.1, Gaussian-window SSIM (Wang et al., 2004) and LDR-FLIP (Andersson et al., 2020), each against Full and GT. The same positions are used for candidate and fixed controls, and for the shifted-mask audit; they need not balance ages under that different mask. Scores do not select the renderer parameters or fixed weight.

Against fixed blending on original-mask Arena, Full-reference LPIPS decreases from 0.00920 to 0.00876 and FLIP from 0.01849 to 0.01751. GT FLIP slightly improves. Appendix .1 reports SSIM for this cohort and the Subway comparisons.

Under the shifted mask, GT LPIPS against fixed blending decreases from 0.16959 to 0.16928 on Arena and from 0.22407 to 0.22401 on Subway. Appendix .1 reports the completed sampled comparisons.

4.5 Refresh phase

To stress-test transfer from refresh observations, we shift informative refreshes by one frame (from positions $8, 16, \ldots$ to $9, 17, \ldots$). Because the loader alternates `NoWarp`/`Warp` inputs at even/odd

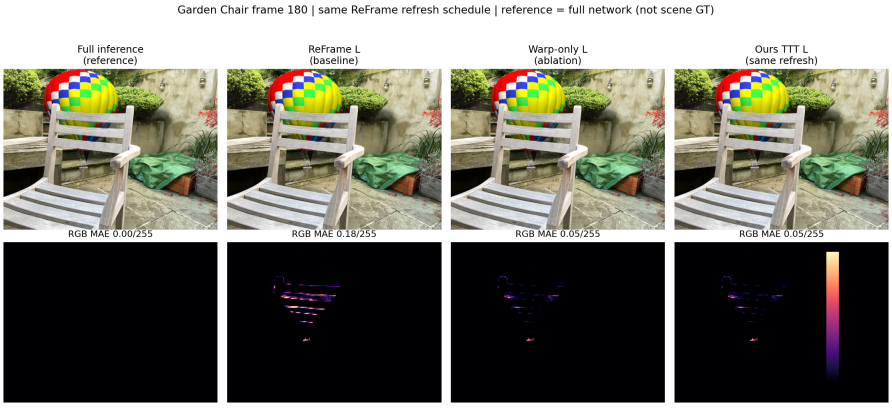

Figure 4: Top row: full inference reference, ReFrame baseline, warp-only ablation, RenderTTT. Middle row: virtual-object crop showing the chair seat where warp's disocclusion artifact is largest. Bottom row: RGB MAE error maps (range 0–32/255). In that crop, RenderTTT matches warp at 0.054/255 MAE and is below ReFrame at 0.178/255.

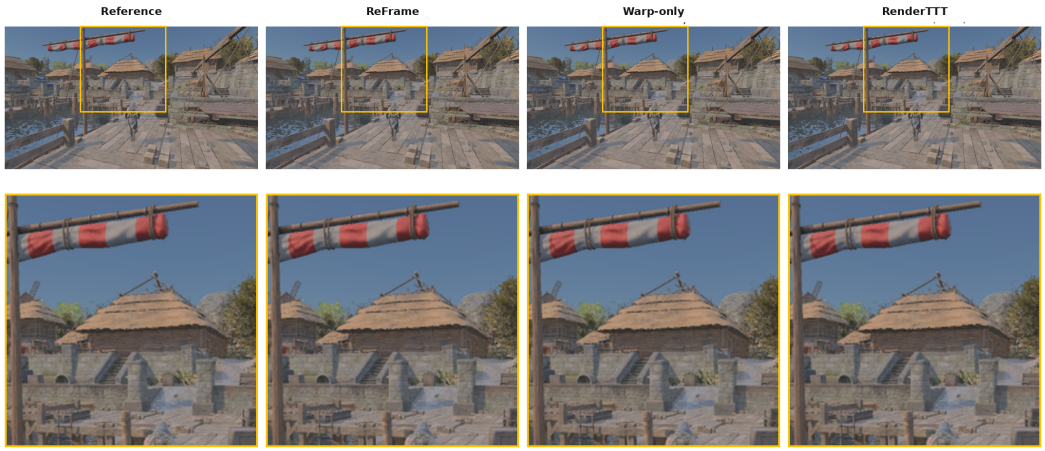

Figure 5: **Qualitative comparison on the fe_medieval rollout (frame 339).** Columns share the same refresh schedule; reference is the frozen full network. The lower row magnifies the yellow box on the banner and roofline.

frames, this shifted mask alters both the input construction seen at refreshes and the initial gap length. Under this shift, spatial correction consistently preserves its Full and GT gains over fixed blending on both scenes, whereas the legacy tile mixture degrades significantly due to negative transfer.Comparing online adaptation against an initial-fit-only control validates the necessity of continued updates: freezing the channel map after the first frame yields lower Full and GT fidelity across reuse frames, confirming that ongoing refresh fitting is essential for tracking feature drift.Finally, extended evaluations on complete sequences under the output-adjacent cache placement confirm that RenderTTT variants consistently outperform native warp in reconstruction fidelity while simultaneously reducing mean and tail latency (Table 5, Table 4).

5 Discussion and Conclusion

When can refresh supervision help? For $e_t = Y_t - B_t$ and correction $c_t = \mathrm{clip}_{[-b,b]}(d_t)$, the squared feature error reduction is

$$|e_t|_2^2 - |e_t - c_t|_2^2 = 2\langle e_t, c_t\rangle - |c_t|_2^2. \tag{10}$$

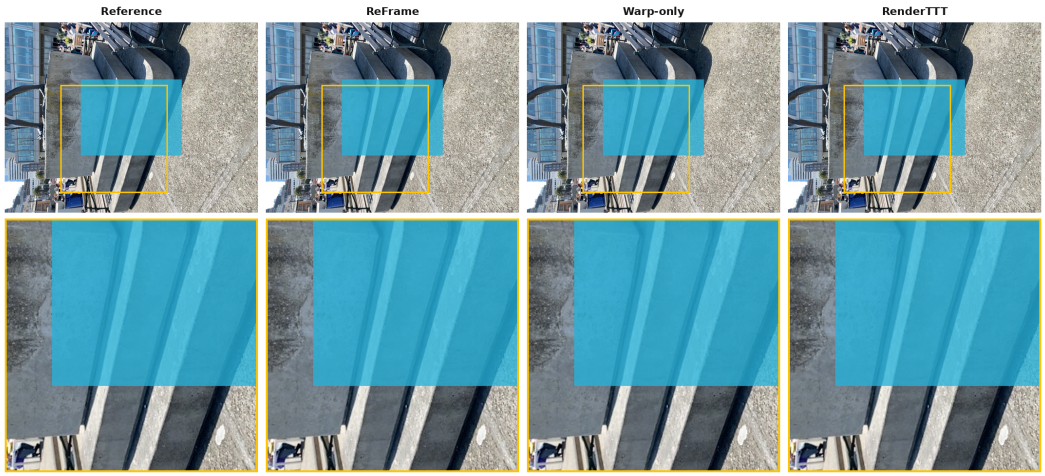

Figure 6: **Playback scan on gandhi** Columns: frozen full-network reference, ReFrame passive reuse, warp-only ablation, RenderTTT. The lower row magnifies the yellow box on the cyan virtual plate. The four source frames differ by at most about two gray levels, so the zoom does not show a shading change.

Improvement requires alignment with the residual $\langle e_t, c_t \rangle$ beyond magnitude clipping. Current shallow features provide spatial observations, while refresh-fitted coefficients steer the deep estimate; remaining residual-transport and clipping terms follow Equation (9).

Reference fidelity vs. target quality. With $\delta = \widehat{I} - I^{\text{full}}$ and $e_{\text{GT}} = I^{\text{full}} - I^{\text{GT}}$, target error decomposes as

$$|\widehat{I} - I^{\text{GT}}|_2^2 - |I^{\text{full}} - I^{\text{GT}}|_2^2 = |\delta| * 2^2 + 2\langle e * \text{GT}, \delta \rangle. \tag{11}$$

Because reducing approximation error does not dictate its interaction with frozen network error, Full and GT fidelity require separate measurement. Empirically, spatial correction reduces both Full and GT MSE across Arena and Subway masks.

Core findings At exact refresh ($q_t = Q, T_t = I$), correction collapses to zero, recovering the exact anchor regardless of regression error. Between refreshes, current-input spatial correction consistently outperforms fixed blending in Full and GT PSNR, while initial-fit-only controls validate the necessity of continued online adaptation. On the pooled Arena and Subway comparisons, spatial correction also improves Full and GT PSNR over native warp and the legacy tile readouts, at lower mean latency. The single-frame panels are a separate record and do not revise those pools. Hyperparameters transfer across scenes and masks without per-scene tuning.

Practical deployment. Deployment hinges on balancing Full/GT fidelity against complete runtime costs. Benchmarked on an A100: RenderTTT-spatial achieves higher Full and GT PSNR than fixed blending, native warp, and legacy tile readouts, while yielding the lowest mean and P95 latency. Spatial correction cuts peak allocation to 428.50/432.10 MiB (versus 484.09/484.30 MiB for baselines); output-adjacent RenderTTT-mixture reaches the highest Full-ref PSNR and lowest reserved memory on Arena (Table 4).

AI USE STATEMENT

Generative AI assisted primarily with wiring and code implementation, formula checking, manuscript restructuring, experiment design, hypothesis refinement, literature lookup, result interpretation, and the creation of vector diagrams and plotting code. AI-assisted scripts were also used to execute and server rendering experiments.

REFERENCES

Pontus Andersson, Jim Nilsson, Tomas Akenine-Möller, Magnus Oskarsson, Kalle Åström, and Mark D. Fairchild. FLIP: A Difference Evaluator for Alternating Images. Proceedings of the ACM on Computer Graphics and Interactive Techniques (HPG), 3(2):1–23, 2020. https://doi.org/10.1145/3406183.

Karlis Martins Briedis, Abdelaziz Djelouah, Mark Meyer, Ian McGonigal, Markus Gross, and Christopher Schroers. Neural Frame Interpolation for Rendered Content. ACM Transactions on Graphics (SIGGRAPH Asia), 40(6), 2021.

Chakravarty R. Alla Chaitanya, Anton S. Kaplanyan, Christoph Schied, Marco Salvi, Aaron Lefohn, Derek Nowrouzezahrai, and Timo Aila. Interactive Reconstruction of Monte Carlo Image Sequences Using a Recurrent Denoising Autoencoder. ACM Transactions on Graphics (SIGGRAPH), 36(4), 2017.

Jiawen Chen, Andrew Adams, Neal Wadhwa, and Samuel W. Hasinoff. Bilateral Guided Upsampling. ACM Transactions on Graphics (SIGGRAPH Asia), 35(6), 2016.

Yossi Gandelsman, Yu Sun, Xinlei Chen, and Alexei A. Efros. Test-Time Training with Masked Autoencoders. NeurIPS, 2022. https://proceedings.neurips.cc/paper_files/paper/2022/file/bcdec1c2d60f94a93b6e36f937aa0530-Paper-Conference.pdf.

Michaël Gharbi, Jiawen Chen, Jonathan T. Barron, Samuel W. Hasinoff, and Frédo Durand. Deep Bilateral Learning for Real-Time Image Enhancement. ACM Transactions on Graphics (SIGGRAPH), 36(4), 2017. https://doi.org/10.1145/3072959.3073592.

Jie Guo, Xihao Fu, Liqiang Lin, Hengjun Ma, Yanwen Guo, Shiqiu Liu, and Ling-Qi Yan. ExtraNet: Real-time Extrapolated Rendering for Low-latency Temporal Supersampling. ACM Transactions on Graphics (SIGGRAPH Asia), 2021.

Kaiming He, Jian Sun, and Xiaoou Tang. Guided Image Filtering. ECCV, pp. 1–14, 2010. Extended version: IEEE Transactions on Pattern Analysis and Machine Intelligence, 35(6):1397–1409, 2013.

Ruian He, Shili Zhou, Yuqi Sun, Ri Cheng, Weimin Tan, and Bo Yan. Low-Latency Space-Time Supersampling for Real-Time Rendering. AAAI, 2024.

Max Jaderberg, Karen Simonyan, Andrew Zisserman, and Koray Kavukcuoglu. Spatial Transformer Networks. NeurIPS, 2015. https://proceedings.neurips.cc/paper_files/paper/2015/file/33ceb07bf4eeb3da587e268d663aba1a-Paper.pdf.

Johannes Kopf, Michael F. Cohen, Dani Lischinski, and Matt Uyttendaele. Joint Bilateral Upsampling. ACM Transactions on Graphics (SIGGRAPH), 26(3), 2007.

Lufei Liu and Tor M. Aamodt. ReFrame: Layer Caching for Accelerated Inference in Real-Time Rendering. ICML, 2025. https://github.com/ubc-aamodt-group/reframe-layer-caching.

Yuejiang Liu, Parth Kothari, Bastien van Delft, Baptiste Bellot-Gurlet, Taylor Mordan, and Alexandre Alahi. TTT++: When Does Self-Supervised Test-Time Training Fail or Thrive? NeurIPS, 2021.

Jiacheng Liu, Chang Zou, Yuanhuiyi Lyu, Junjie Chen, and Linfeng Zhang. From Reusing to Forecasting: Accelerating Diffusion Models with TaylorSeers. ICCV, 2025. https://openaccess.thecvf.com/content/ICCV2025/html/Liu_From_Reusing_to_Forecasting_Accelerating_Diffusion_Models_with_TaylorSeers_ICCV_2025_paper.html.

Xuan Luo, Jia-Bin Huang, Richard Szeliski, Kevin Matzen, and Johannes Kopf. Consistent Video Depth Estimation. ACM Transactions on Graphics (SIGGRAPH), 39(4), 2020. https://doi.org/10.1145/3386569.3392377.

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. ICLR, 2025. https://proceedings.iclr.cc/paper_files/paper/2025/hash/518046d86bbc41a0707727c38301ad8e-Abstract-Conference.html.

Xinyin Ma, Gongfan Fang, and Xinchao Wang. DeepCache: Accelerating Diffusion Models for Free. CVPR, 2024.

Xinyin Ma, Gongfan Fang, Michael Bi Mi, and Xinchao Wang. Learning-to-Cache: Accelerating Diffusion Transformer via Layer Caching. NeurIPS, 2024. https://proceedings.neurips.cc/paper_files/paper/2024/hash/f0b1515be276f6ba82b4f2b25e50bef0-Abstract-Conference.html.

Ravi Teja Mullapudi, Steven Chen, Keyi Zhang, Deva Ramanan, and Kayvon Fatahalian. Online Model Distillation for Efficient Video Inference. ICCV, 2019. https://openaccess.thecvf.com/content_ICCV_2019/html/Mullapudi_Online_Model_Distillation_for_Efficient_Video_Inference_ICCV_2019_paper.html.

Thomas Müller, Fabrice Rousselle, Jan Novák, and Alexander Keller. Real-time Neural Radiance Caching for Path Tracing. ACM Transactions on Graphics, 2021. https://doi.org/10.1145/3450626.3459812.

Umair Nawaz, Ahmed Heakl, Ufaq Khan, Abdelrahman M. Shaker, Salman Khan, and Fahad Shahbaz Khan. WorldCache: Content-Aware Caching for Accelerated Video World Models. arXiv:2603.22286, 2026. https://arxiv.org/abs/2603.22286.

Diego Nehab, Pedro V. Sander, Jason Lawrence, Natalya Tatarchuk, and John R. Isidoro. Accelerating Real-Time Shading with Reverse Reprojection Caching. Graphics Hardware, 2007.

Shuaicheng Niu, Jiaxiang Wu, Yifan Zhang, Yaofo Chen, Shijian Zheng, Peilin Zhao, and Mingkui Tan. Efficient Test-Time Model Adaptation without Forgetting. ICML, 2022. https://proceedings.mlr.press/v162/niu22a.html.

Shuaicheng Niu, Jiaxiang Wu, Yifan Zhang, Zhiquan Wen, Yaofo Chen, Peilin Zhao, and Mingkui Tan. Towards Stable Test-Time Adaptation in Dynamic Wild World. ICLR, 2023. https://openreview.net/forum?id=g2YraF75Tj.

Peter O'Connor and Max Welling. Sigma Delta Quantized Networks. ICLR, 2017. https://openreview.net/forum?id=HkNRsU5ge.

Mathias Parger, Chengcheng Tang, Christopher D. Twigg, Cem Keskin, Robert Wang, and Markus Steinberger. DeltaCNN: End-to-End CNN Inference of Sparse Frame Differences in Videos. CVPR, 2022.

Mathias Parger, Chengcheng Tang, Thomas Neff, Christopher D. Twigg, Cem Keskin, Robert Wang, and Markus Steinberger. MotionDeltaCNN: Sparse CNN Inference of Frame Differences in Moving Camera Videos. ICCV, 2023.

Daniel Scherzer, Lei Yang, Oliver Mattausch, Diego Nehab, Pedro V. Sander, Michael Wimmer, and Elmar Eisemann. Temporal Coherence Methods in Real-Time Rendering. Computer Graphics Forum, 31(8):2378–2408, 2012.

Imanol Schlag, Kazuki Irie, and Jürgen Schmidhuber. Linear Transformers Are Secretly Fast Weight Programmers. ICML, 2021. https://proceedings.mlr.press/v139/schlag21a.html.

Steffen Schneider, Evgenia Rusak, Luisa Eck, Oliver Bringmann, Wieland Brendel, and Matthias Bethge. Improving Robustness against Common Corruptions by Covariate Shift Adaptation. NeurIPS, 2020. https://proceedings.neurips.cc/paper/2020/hash/85690f81aadc1749175c187784afc9ee-Abstract.html.

Evan Shelhamer, Kate Rakelly, Judy Hoffman, and Trevor Darrell. Clockwork Convnets for Video Semantic Segmentation. ECCV Workshops, 2016.

Zhirong Shen, Rui Huang, Chang Zou, Shikang Zheng, Jiacheng Liu, Peiliang Cai, Zhengyi Shi, Yaosong Du, Liang Feng, Xiaobing Tu, Jinkui Ren, Xiantao Zhang, and Linfeng Zhang. Accelerating Diffusion Transformers with Gaussian Process Rectified Feature Cache. ECCV, LNCS 17041, pp. 320–337, 2026. https://doi.org/10.1007/978-3-032-37429-5_19.

Karen Simonyan and Andrew Zisserman. Very Deep Convolutional Networks for Large-Scale Image Recognition. ICLR, 2015. https://arxiv.org/abs/1409.1556.

Yu Sun, Xiaolong Wang, Zhuang Liu, John Miller, Alexei A. Efros, and Moritz Hardt. Test-Time Training with Self-Supervision for Generalization under Distribution Shifts. ICML, 2020.

Yu Sun, Xinhao Li, Karan Dalal, Jiarui Xu, Arjun Vikram, Genghan Zhang, Yann Dubois, Xinlei Chen, Xiaolong Wang, Sanmi Koyejo, Tatsunori Hashimoto, and Carlos Guestrin. Learning to (Learn at Test Time): RNNs with Expressive Hidden States. ICML, 2025.

Narayanan Sundaram, Thomas Brox, and Kurt Keutzer. Dense Point Trajectories by GPU-Accelerated Large Displacement Optical Flow. ECCV, pp. 438–451, 2010.

Zachary Teed and Jia Deng. RAFT: Recurrent All-Pairs Field Transforms for Optical Flow. ECCV, 2020. https://doi.org/10.1007/978-3-030-58536-5_24.

Manu Mathew Thomas, Gabor Liktor, Christoph Peters, Sungye Kim, Karthik Vaidyanathan, and Angus G. Forbes. Temporally Stable Real-Time Joint Neural Denoising and Supersampling. Proceedings of the ACM on Computer Graphics and Interactive Techniques (HPG), 5(3), Article 21, 2022. https://doi.org/10.1145/3543870.

Aladin Virmaux and Kevin Scaman. Lipschitz Regularity of Deep Neural Networks: Analysis and Efficient Estimation. NeurIPS, 2018. https://proceedings.neurips.cc/paper/2018/hash/d54e99a6c03704e95e6965532dec148b-Abstract.html.

Bruce Walter, George Drettakis, and Steven Parker. Interactive Rendering using the Render Cache. Eurographics Workshop on Rendering, 1999.

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.

Dequan Wang, Evan Shelhamer, Shaoteng Liu, Bruno Olshausen, and Trevor Darrell. Tent: Fully Test-Time Adaptation by Entropy Minimization. ICLR, 2021.

Renhao Wang, Yu Sun, Arnuv Tandon, Yossi Gandelsman, Xinlei Chen, Alexei A. Efros, and Xiaolong Wang. Test-Time Training on Video Streams. Journal of Machine Learning Research, 26, 2025. https://jmlr.org/papers/v26/24-0439.html.

Songyin Wu, Sungye Kim, Zheng Zeng, Deepak Vembar, Sangeeta Jha, Anton Kaplanyan, and Ling-Qi Yan. ExtraSS: A Framework for Joint Spatial Super Sampling and Frame Extrapolation. SIGGRAPH Asia, 2023.

Lei Xiao, Salah Nouri, Matt Chapman, Alexander Fix, Douglas Lanman, and Anton Kaplanyan. Neural Supersampling for Real-time Rendering. ACM Transactions on Graphics (SIGGRAPH), 2020.

Zhikang Xie, Xichen Ye, Yifan Wu, Haoshen Yu, Chenan Li, Peizhu Gong, Weizhong Zhang, and Cheng Jin. OnlineCache: Learning Dynamic Caching Policies with Error Correction for Efficient Diffusion Inference. arXiv:2607.29398, 2026.

Frank Yu, Sid Fels, and Helge Rhodin. Scaling Neural Face Synthesis to High FPS and Low Latency by Neural Caching. WACV, 2023.

Richard Zhang, Phillip Isola, Alexei A. Efros, Eli Shechtman, and Oliver Wang. The Unreasonable Effectiveness of Deep Features as a Perceptual Metric. CVPR, 2018. https://openaccess.thecvf.com/content_cvpr_2018/html/Zhang_The_Unreasonable_Effectiveness_CVPR_2018_paper.html.

Marvin Zhang, Sergey Levine, and Chelsea Finn. MEMO: Test Time Robustness via Adaptation and Augmentation. NeurIPS, 2022. https://proceedings.neurips.cc/paper_files/paper/2022/hash/fc28053a08f59fccb48b11f2e31e81c7-Abstract.html.

Xizhou Zhu, Yuwen Xiong, Jifeng Dai, Lu Yuan, and Yichen Wei. Deep Feature Flow for Video Recognition. CVPR, 2017.

Jiafan Zhuang, Zilei Wang, and Bingke Wang. Video Semantic Segmentation with Distortion-Aware Feature Correction. IEEE Transactions on Circuits and Systems for Video Technology, 31(8):3128–3139, 2021. https://doi.org/10.1109/TCSVT.2020.3037234.

Chang Zou, Xuyang Liu, Ting Liu, Siteng Huang, and Linfeng Zhang. Accelerating Diffusion Transformers with Token-wise Feature Caching. ICLR, 2025. https://proceedings.iclr.cc/paper_files/paper/2025/hash/bbe024e0517fe12ac3a8d388b19ff9fe-Abstract-Conference.html.

.1 SPATIAL READOUT DETAILS

The spatial method uses a single global 24×24 matrix, fitted from within-frame centered shallow/deep correspondences at every refresh, including initialization. Shallow pooling, motion composition, age blending and clipping are included in the measured model call. Its parameters are unchanged across the two scenes and the two masks. Original refresh positions are $0, 8, 16, \ldots$; shifted positions are $0, 9, 17, \ldots$. Both masks have 16 refreshes on Arena and 125 on Subway. The first shifted interval has length nine rather than eight. The spatial method retains two $1 \times 24 \times 272 \times 480$ FP32 anchors (23.90625 MiB together) and three 24×24 matrices (6.75 KiB together), in addition to motion and current-feature state. These logical tensor sizes exclude temporary workspaces and are not allocator peaks. Spatial runs achieve lower peak allocated memory than their fixed controls on both original-mask scenes.

The initial-fit-only control preserves the same first-frame moments and coefficients, then updates only the shallow/deep anchors. Recorded state hashes are constant at all later refreshes and reuse frames, and predictions agree with the online method until its first subsequent refresh. On original-mask Arena, initial-only Full/GT PSNR is 34.183/30.126 dB, compared with online 34.688/30.272 dB. On shifted-mask Arena, initial-only reaches 33.876/29.501 dB versus online 34.255/29.608 dB. These development-prefix controls show the benefit of continued fitting; they are not timing or perceptual measurements.

Saved-pixel and perceptual protocol. Each perceptual cohort replays the unchanged method and fixed control separately from timing. All frame errors must reproduce their measured records, and sample RGB hashes must match the timed candidate fingerprints. Lossless candidate, fixed, Full and GT arrays are exported for independent pixel-error checks. The original fixed-control images and metrics reproduce the previously archived cohort. LPIPS-VGG v0.1 uses normalized inputs; SSIM uses a Gaussian window with $\sigma = 1.5$, population covariance and unit range; LDR-FLIP uses 67 pixels per degree.

.2 TILE-BASED READOUT DETAILS

The primary comparison uses one 24-channel STSS cache at the output of `up2`, while recomputing current shallow features and `up3` on every frame. The secondary output-adjacent experiments instead cache the output of `up3`, immediately before the output head. The two placements are identified separately throughout. Released scene-specific weights remain fixed. The native-motion backend composes displacements on CUDA; the RAFT backend estimates motion on CUDA. Both retain the same refresh-before-replacement ordering and use bilinear feature sampling. The tile grid is 6×8, coefficient EMA decay is 0.5 and the affine residual bound is 0.4 times channel magnitude. RAFT affine uses the declared consistency gate; native age-affine disables that gate. The mixture uses a target-energy-normalized ridge prior of 0.02 toward warp and coefficients restricted to $[0, 1]$.

Source and result provenance. The source bundle retains checkpoint and source hashes, complete per-frame inference traces, fixed development choices and raw evaluation records. Paper tables select the configurations from these records. Arena covers IDs 500–1500; Lewis covers 5–1000. Development prefixes, complete sequences and partial follow-up work are explicitly distinguished. The full-network evaluation target never enters a reuse model call. Pending scene or perceptual cohorts supply no numbers to the paper.

Causal state. At refresh, the old anchor is transported, fitted state is updated from the fresh feature, and the fresh feature becomes the anchor. At reuse, fitted coefficients and the anchor remain unchanged; the motion accumulator and cache-age counter advance. The current-image bilinear residual remains active at both sites. Current shallow features and the final decoder are recomputed on primary branch reuse; they are skipped in the secondary output-adjacent placement. Reset clears both cached activation and adaptive state. These legacy tile methods perform no fit at initialization; the spatial method in Appendix .1 fits immediately.

Baseline topology and scope. ReFrame's U-Net construction caches the deep input to the final decoder block and recomputes both its shallow skip features and that decoder block. Our primary STSS adaptation follows this pattern: retain the output of `up2`, recompute `conv1`/`conv2`, and

evaluate `up3` on every frame. The secondary output-adjacent placement reuses the output of `up3` and evaluates only the output head plus current-image residual. The comparison in Appendix B controls this distinction explicitly.

A Output Magnitude Bounds

For an affine correction clipped elementwise to $[-\kappa s_c, \kappa s_c]$ in channel c, and age scale $\eta_t \in [0, 1]$, the feature perturbation satisfies

$$\|\widehat{Y} - Z\|_F^2 \le \eta_t^2 HW\kappa^2 \sum_c s_c^2. \tag{12}$$

If the frozen output network is locally L-Lipschitz on the segment between the two features, the output change is at most $L\eta_t\kappa\sqrt{HW\sum_c s_c^2}$. We do not estimate L: its exact computation is NP-hard even for two-layer networks, and standard bounds can overestimate it substantially (Virmaux and Scaman, 2018). The mixture is coordinatewise between passive and transported anchors: its passive weights satisfy $0 \le \alpha_i \le \eta_t$, giving $\|\widehat{Y} - Z\|_F \le \eta_t \|A - Z\|_F$. Spatially varying weights need not place the feature vector on a single line segment.

B Primary Cache Controls and Passive Policies

This section uses the primary branch cache: the output of `up2`, with current `conv1`/`conv2` features and `up3` decoding on every frame. The output-adjacent experiments are a distinct cache placement. The primary Arena cohort comprises IDs 500–627; Subway comprises IDs 6–1000. Every result uses the same scene-specific checkpoint, ordered inputs and immutable full-network references as its matched controls. ReFrame N-8 at this site is a fresh passive run with the same measurement protocol as the adaptive readouts.

Fixed comparison before transfer. The eight-point grid below is evaluated only on the Arena development prefix. The criterion is pooled clipped-RGB MSE against the full network, including refresh frames; ties favor the smaller coefficient. It selects $\lambda = 1$ before dispatching any Subway branch run. Each Subway process verifies the saved selection record. At $\lambda = 0$, the fixed readout agrees with warp. At $\lambda = 1$, its age-scaled passive weight varies from $1/8$ to $7/8$ after an informative N-8 refresh, so it remains distinct from the passive anchor. The first interval uses warp for the legacy tile and age-scaled fixed readouts. Spatial correction instead starts from its initial fit, with warp as its base during that interval. This coefficient is unrelated to the secondary placement's earlier choice of 0.1.

Table 2: **Primary-site fixed grid on Arena.** All 128 development frames, N-8, ten warmup triplets. The selected weight minimizes Full-reference MSE; bold marks selection. Allocated and reserved peaks (MiB) include benchmark tensors.

λ	Full (dB)	GT (dB)	Mean (ms)	P95 (ms)	Alloc.	Res.
0	33.947	30.001	5.343	10.112	484.09	626.00
0.01	33.949	30.002	5.338	10.132	484.09	626.00
0.025	33.951	30.004	5.366	10.158	484.09	626.00
0.05	33.956	30.007	5.336	10.117	484.09	626.00
0.1	33.964	30.012	5.358	10.129	484.09	626.00
0.25	33.990	30.028	5.335	10.105	484.09	626.00
0.5	34.029	30.055	5.363	10.094	484.09	626.00
1	**34.068**	**30.092**	**5.354**	**10.121**	**484.09**	**626.00**

Warmup and readout cost. All primary N-8 rows use a fresh process and ten warmup triplets, each exercising refresh, reuse, and a further refresh with fitting. State is then reset before measurement. Mean/P95 include the first interval and every refresh. An older shallow-policy cohort uses ten refresh/reuse pairs and then resets state; its full N-8 quality is identical to the matched passive

rerun, but we retain the actual timings from each protocol. N-2/N-4 in the main table come from that policy cohort.

The branch anchor has shape $1 \times 24 \times 272 \times 480$ in FP32, or 11.953125 MiB. The two affine coefficient arrays occupy 9 KiB and the mixture array occupies 4.5 KiB. These are logical state sizes, not allocator peaks: native motion retains additional state, and applying the readout constructs dense fields and activations. Peak allocated and reserved memory are measured from the full process, including the benchmark's evaluation tensors. The legacy matched N-8 runs in this section reserve 626 MiB.

Additional passive refresh policies. The following cohort retains the same branch cache. Periodic N-2/N-4/N-8 refresh by sequence position, including the first frame. The frame-delta controls use the established input-change score and strict threshold comparison, now with current shallow features and final decoding. Thresholds 20 and 25 are fixed rather than tuned per scene. The table reports their actual refresh counts because a high reference-fidelity score can result from many more full evaluations.

Table 3: **Additional passive policies at the branch cache.** Ten warmup pairs, one process per configuration. R counts refreshes; $\Delta 20/25$ apply the fixed strict input-change thresholds. Times are milliseconds and peaks are MiB. **Bold** marks the best and underline the second-best value per column within each scene. These N-8 times belong to the policy cohort, not the matched-triplet rerun in the main table.

Policy	R	Full (dB)	GT (dB)	Mean	P95	Alloc.	Res.
Arena: 128 frames							
N-2	64	34.293	**30.121**	6.715	9.486	**468.46**	626.00
N-4	32	34.093	30.092	5.308	9.457	**468.46**	626.00
N-8	16	33.974	30.100	**4.647**	9.457	**468.46**	626.00
$\Delta 20$	14	**34.823**	30.067	5.006	9.935	480.48	626.00
$\Delta 25$	8	34.565	**30.121**	4.704	**8.434**	480.48	626.00
Subway: 995 frames							
N-2	498	38.632	28.612	6.706	9.492	**468.46**	626.00
N-4	249	37.266	28.504	5.346	**9.460**	**468.46**	626.00
N-8	125	36.346	28.397	**4.663**	9.466	**468.46**	626.00
$\Delta 20$	819	**49.919**	**28.916**	8.948	9.951	480.48	626.00
$\Delta 25$	761	47.219	28.891	8.620	9.962	480.48	626.00

C MEASUREMENT PROTOCOL

All inference experiments use one A100-SXM4-40GB, released STSS checkpoints, FP32 and disabled TF32/cuDNN autotuning. Each configuration runs in a fresh process with state reset after warmup. Primary N-8 rows use ten refresh/reuse/refresh triplets; the additional passive-policy cohort uses ten refresh/reuse pairs. CUDA synchronization brackets the model call (motion composition, transport, fitting, readout); dataset decoding, transfers and scoring are excluded. Following ReFrame (Liu and Aamodt, 2025), we distinguish fidelity, mean/P95 latency and memory. A100 timings are not directly comparable with ReFrame's RTX 2080 Ti measurements.

Memory accounting. Allocated peaks include model parameters, inputs, cache state, activations and the benchmark target. Reserved peaks additionally include allocator and graph-pool storage. The primary branch anchor occupies 11.95 MiB; the output-adjacent anchor occupies 47.81 MiB. Coefficient arrays occupy 9 KiB (affine) and 4.5 KiB (mixture). All measured peaks include the live evaluation target and are benchmark-process figures.

Independent record checks. Exact frame IDs, refresh masks, checkpoint/source hashes, raw per-frame times, errors and memory peaks are checked before reporting completion. Pooled MSE and PSNR, mean/P95 time and global peaks are recomputed from raw traces.

D OUTPUT-ADJACENT FULL-SEQUENCE RESULTS

All configurations complete the same 1,001 Arena frame IDs on the A100. We compare ReFrame-style N-8 caching, stronger N-2/N-4 refresh schedules, matched transport and correction. Full-network and dataset-GT references are reported separately.

Table 4: Complete Arena audit. Mean/P95 are model-call milliseconds; Full-ref/GT are dB. N-8 unless named otherwise. **Bold** marks the best and underline the second-best value per column.

Method	Mean	P95	Alloc.	Res.	Full-ref	GT
Full network	9.87	10.59	**455.39**	626	exact	**31.243**
ReFrame N-8	**1.69**	9.50	503.40	626	30.130	28.259
ReFrame N-4	2.82	9.51	503.40	626	30.635	28.464
ReFrame N-2	5.06	9.53	503.40	626	<u>31.551</u>	28.785
Extrapolation	<u>1.85</u>	9.76	551.40	674	28.571	27.166
RAFT graph warp	6.34	14.18	528.32	680	30.590	28.581
RTT-affine, RAFT (ours)	2.41	9.28	475.20	602	32.415	29.481
RTT-mixture, RAFT (ours)	2.28	9.12	468.50	578	32.528	29.585
Renderer MV warp	2.73	10.43	523.37	674	32.152	29.281
RenderTTT-affine (ours)	2.22	<u>8.85</u>	<u>462.10</u>	<u>555</u>	<u>32.685</u>	29.650
RenderTTT-mixture (ours)	<u>2.18</u>	**8.52**	458.30	**540**	**32.852**	<u>29.765</u>

Renderer-motion warp reaches 32.152 dB at a mean of 2.73 ms (P95: 10.43 ms) versus ReFrame N-2's 31.551 dB at 5.06 ms. While renderer motion provides a strong starting point, our RenderTTT variants significantly improve upon it in both quality and efficiency. Specifically, RenderTTT-affine and RenderTTT-mixture improve over the matched warp baseline by $+0.533$ dB and $+0.700$ dB in Full-ref PSNR. Crucially, this quality gain is accompanied by a reduction in computational cost: their mean latencies drop to 2.22 ms and 2.18 ms (a $\sim$19–20% speedup), and P95 latencies decrease to 8.85 ms and 8.52 ms, demonstrating superior stability and throughput.

E CROSS-SCENE FULL-SEQUENCE RESULTS

The following tables report the full-sequence outcomes for Lewis, SunTemple and Subway at the output-adjacent cache. Each scene uses its released scene-specific checkpoint with one timed run per configuration.

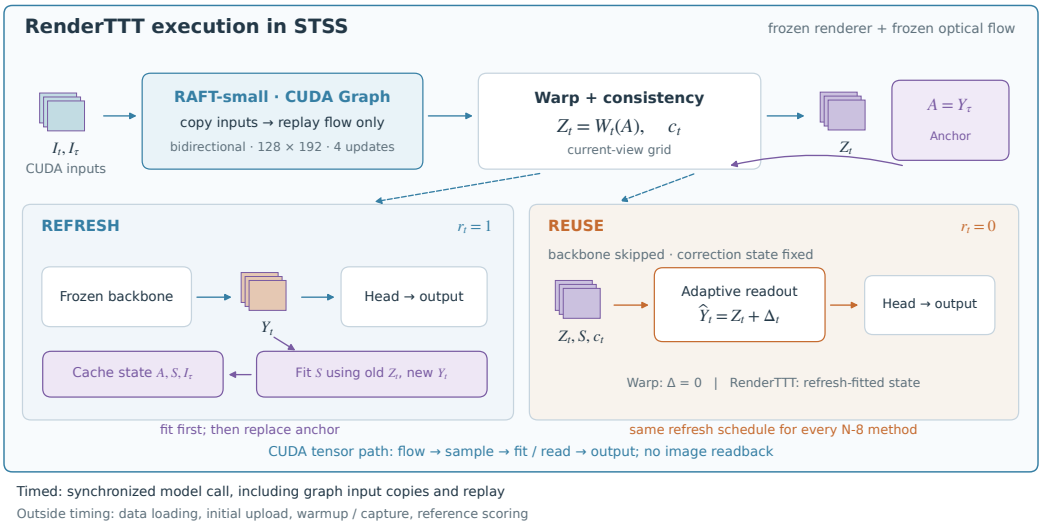

Figure 7: Measured STSS implementation. The fixed-shape RAFT estimator alone is captured by CUDA Graph; the renderer, resampling and adaptive readout execute on CUDA outside that capture. Refresh frames fit correction state using the transported old anchor and newly computed activation, then replace the anchor. Reuse frames skip the backbone and keep correction state fixed.

Table 5: Complete Subway inference. **Bold** marks the best and underline the second-best per column.

	PSNR↑		Time↓		Peak↓	
Readout	Full	GT	Mean	P95	Alloc.	Res.
Full network	—	**28.939**	9.928	12.363	**455.39**	**626.00**
ReFrame N-8	31.659	27.290	**1.687**	9.526	503.40	**626.00**
ReFrame N-4	32.413	27.484	2.803	9.536	503.40	**626.00**
ReFrame N-2	34.080	27.852	5.050	**9.524**	503.40	**626.00**
RAFT warp	32.623	27.556	6.346	14.185	528.32	680.00
RTT-mixture, RAFT (ours)	33.146	27.722	7.165	17.883	528.50	776.00
Native warp	34.352	27.981	2.721	10.391	523.37	674.00
RTT-mixture, native (ours)	**34.731**	28.067	3.808	14.158	523.38	772.00

Subway (995 frames). Native mixture adds 0.379/0.086 dB over warp against full/GT, while increasing mean time by 1.087 ms and reserved peak by 98 MiB.

Independent verification. For each scene we reaggregate all per-frame records (e.g. 10,945 frames for Subway), recompute reference errors from saved RGB images, verify ordered IDs, refresh masks, source/checkpoint hashes and perceptual row means. The remaining frames are checked through recorded errors. Raw traces, image hashes and audit records accompany the source bundle.

F OUTPUT-ADJACENT CACHE PLACEMENT

The following table contains the output-adjacent experiments. They cache the output of `up3`, evaluate the output head and current-image residual on reuse, and skip current learned shallow features and final decoding. Their ReFrame labels denote that shared-site passive adaptation, not the stronger branch baseline in the primary comparison. The complete Arena rollout has 1,001 frames; its results are reported separately from the 128-frame development prefix. Lewis and SunTemple contain 996 frames each, and Subway contains 995.

The age-conditioned and cross-scene diagnostic figures from earlier drafts of this appendix are omitted: on the secondary placement the affine and mixture readouts do not consistently improve over renderer-motion warp, and the diagnostic panels would report that result without an independent corrective signal.

G TILE-BASED READOUTS: IMPLEMENTATION NOTES

This appendix section collects implementation details for the affine and mixture readouts that are referenced from Section 3. The current-input spatial method is described in that section.

Causal execution and computational cost. On a refresh: compute Y, compare it with the transported *old* anchor, update state, replace the anchor with Y, and emit the full-network result. On reuse: compute current shallow features, compose motion from available inputs, transport the anchor, read the frozen correction state, and evaluate the final decoder and output head. The current shallow skip and RGB residual are therefore present for every primary readout, including passive caching. A reset clears correction state together with the anchor.

For C channels and a $H \times W$ feature map, the anchor requires $O(CHW)$ state; affine or mixture coefficients add $O(CT)$ state for $T = 48$ grid cells. Applying a readout still touches $O(CHW)$ values. Moment fitting requires pooled first and second moments, with scalar divisions rather than a dense matrix inverse. A compact fit does not make motion or tensor operations free: displacement composition or flow inference, resampling, coefficient interpolation and the remaining network layers all contribute to latency. If p is the refresh fraction, the mean latency is approximately

$$\overline{T} = p(T_{\text{full}} + T_{\text{update}}) + (1-p)(T_{\text{tail}} + T_{\text{read}}) + T_{\text{flow}} + T_{\text{sample}}. \tag{13}$$

Here T_{tail} includes the current shallow features and final decoder in the primary architecture. Flow and sampling costs are amortized per frame. Fixed p controls deep-branch invocation count, not latency. Section 4.2 measures this balance.

Transport and confidence. The native backend composes current-to-previous renderer motion into current-to-anchor displacement. For one-step motion f_t and accumulated field F_{t-1}, $F_t(p) = f_t(p) + F_{t-1}(p + f_t(p))$. Bilinear sampling propagates the displacement and an image-bound validity field. The alternative RAFT-small backend (Teed and Deng, 2020) estimates forward and backward motion between the anchor and current image at 128×192 resolution, using four updates; it computes consistency on the current grid and forms $c = \exp(-e/2)$ from the displacement disagreement (Sundaram et al., 2010). The fixed-shape estimator alone is captured by CUDA Graph; sampling, fitting and rendering remain outside that graph. Both backends resize displacement to the feature grid and use bilinear sampling (Jaderberg et al., 2015). RAFT affine uses

$$g(c) = \text{clip}\left(\frac{c - c_0}{1 - c_0}, 0, 1\right), \qquad c_0 = 0.35. \tag{14}$$

The native age-affine configuration sets this readout gate to zero, retaining motion-path validity only in its fitting weights. Neither backend's confidence is a calibrated probability of feature correctness.