

Beyond the Brute Force

Pair-Based Screen-Space Rasterization Makes Geometry-First
VR Rendering Fit the Latency Budget at Production Scale

Ryosuke Kawai

ryosukekawai1224@gmail.com

April 2026

Abstract

In a companion paper [1] we presented a prototype of a geometry-first VR rendering architecture based on Structure Graph Data (SGD), and argued that a deterministic per-frame raycaster structurally avoids the specular and disocclusion failure modes of Asynchronous Timewarp (ATW). That paper, however, was cautious about scalability: its renderer operated on a 17-face toy scene at 160×90 resolution, and the gap analysis extrapolated that at 10,000 polygons, even on an NVIDIA A100, a frame would cost roughly 405 ms — “well outside the 83.3 ms” sEMG-prediction budget [1]. Scaled by pixel count to Quest 3 per-eye resolution (2160×1080), that extrapolation implies several seconds per frame: two orders of magnitude over a 60 Hz display budget.

In this paper we show that the bottleneck was not architectural but implementational. The original prototype’s brute-force Möller–Trumbore path tested every pixel against every triangle, producing a cost that scales as $O(H \cdot W \cdot F)$ and dominates once either term grows. We replace this with a *pair-based screen-space SGD rasterizer* (SSSR): triangles are projected to screen, and only the (pixel, triangle) pairs inside each triangle’s screen bounding box are enumerated and tested, with depth resolved by a single scatter-min. This reduces the asymptotic cost to $O(H \cdot W + \sum_f A_f)$, where A_f is the screen area of face f . In a procedurally generated 9,738-face furnished room at 2160×1080 , a single A100 frame completes in **14.47 ms** — a **482×** speedup over the brute-force baseline (6,967 ms), fitting comfortably within the 60 Hz display budget (16.7 ms) and within a three-candidate predictive pre-rendering window of 83 ms. At 4K per eye (3840×2160) the same scene renders in **48.1 ms** (515×). Across five resolutions spanning a $36\times$ range in pixel count, SSSR cost is 5.8–16 μ s per kilopixel, an empirical confirmation

that the architecture is genuinely $O(\text{pixels})$ once the implementation matches the design.

Beyond compute throughput, we also measure rendering correctness (SSSR matches a brute-force oracle at 47.1 dB PSNR / 99.6% face-ID agreement), temporal stability (ATW drifts -14.20 dB from frame 1 to 30 on the same scene, versus SGD which by construction does not drift), mirror-reflection correctness (analytic reflected-camera rasterization matches a brute-force secondary-ray reference at 43.3 dB on mirror pixels after a clip-plane and chirality fix), face-count scalability ($8\times$ more triangles costs only $2.5\times$ more time), and — using FovVideoVDP, a VR-aware perceptual metric — find an SGD-over-ATW perceptual advantage of $+4.75$ JOD, well above the threshold at which VR quality differences are considered clearly visible.

We situate these results carefully. **All measurements in this paper are software-only on an NVIDIA A100 GPU; no frame is ever displayed on a physical HMD, no sEMG electrode is connected, and no Motion-to-Photon measurement is taken with a high-speed camera.** On a shipping VR device, additional latency sources (IMU and sEMG bus transfer, display composition, panel scan-out, pixel persistence) combine with mobile-SoC thermal and power envelopes to produce qualitatively different behavior; our numbers should be read as lower bounds on what is possible, not predictions of what a Quest 3 or Vision Pro will measure. The companion paper’s caveats about surface-EMG physiological validation, production-ATW comparison, real-HMD end-to-end latency, and mobile system-on-chip deployment remain fully open. What changes is the balance: the most conspicuous negative of the prior work — “per-frame geometry rendering will not fit a VR budget at realistic polygon counts” — is no longer supported by our own data, and should be retracted from the prototype’s limitations.

1. Introduction

Our companion paper [1] argued that Asynchronous Timewarp (ATW), the dominant latency-compensation primitive in consumer VR, has three architectural failure modes — disocclusion holes, view-dependent surfaces, and feedback-loop temporal drift — that cannot be eliminated by any refinement of pixel-space warping, and proposed Structure Graph Data (SGD) raycasting as an alternative. That paper is a measurement of the failure modes and a feasibility sketch of the alternative; it is not a scalability study.

On that front the companion paper was quite restrained. The renderer it evaluated operated on a 17-face test scene at 160×90 pixels. Its gap analysis — the section that has since become the paper’s most cited passage — extrapolated: “At 10,000 faces, extrapolating the empirical sub-linear exponent (0.60) predicts $22\times$ the 17-face render time; at $N_f=10,000$ and A100 speed of 18.4 ms for $N_f=17$, this yields approximately 405 ms per frame, well outside the 83.3 ms budget” [1, §6.2]. The text went on to note that consumer VR headsets operate at one to two orders of magnitude lower GPU throughput than an A100, and concluded that production deployment was at minimum an open engineering problem.

That gap analysis was the single most pessimistic aspect of the prototype. It was also the aspect that turned out to be wrong.

What changed. In revisiting the prototype’s renderer with the explicit goal of a hundred-fold reduction in compute cost, we discovered that the original raycaster was fundamentally misaligned with the architecture it was meant to validate. The architectural specification in [2] is explicit that “[t]he unit does not attempt to hallucinate future frames from the current image alone. Instead, it determines which SGD surfaces will be visible from each candidate viewpoint and renders from the local geometric database” — a description whose natural complexity is linear in the number of SGD surfaces and in pixels, not in their product. The reference implementation in the companion paper, however, was a GPU port of Möller–Trumbore triangle intersection in which every pixel tested every face and aggregated via \min , with cost $O(H \cdot W \cdot F)$. At 2160×1080 against 9,738 faces, that is roughly twenty-three billion ray–triangle tests per frame — quite literally the workload the architecture was written to avoid.

The revised design moves the triangle–pixel meeting point out of the primary-ray search and into screen-space enumeration. After projecting all faces to screen, we enumerate the set of (pixel, triangle) pairs for which

the pixel falls inside the triangle’s screen-space bounding box, evaluate an edge-function in-triangle test per pair, and resolve per-pixel depth with a single scatter-min operation across the pair list. The cost is bounded by the sum of screen-space bounding-box areas, $\sum_f A_f$, which for a typical furnished room is on the order of a few $H \cdot W$ regardless of F . Empirically, the per-kilopixel cost on an A100 is 5.8–16 μs across a $36\times$ range of pixel counts, with the implicit F -dependence absorbed into screen coverage rather than into a multiplicative factor.

Contributions. This short paper reports three results:

1. A diagnosis of why the companion paper’s scalability numbers understated the architecture’s potential, and a precise decomposition of where the cost was going.
2. A GPU-friendly pair-based rasterizer for SGD scenes, with explicit memory budgeting that runs a 9,738-face furnished room at 2160×1080 in 14.47 ms on an A100, and at 3840×2160 in 48.1 ms.
3. A resolution sweep from 640×360 to 4K-per-eye that empirically confirms $O(\text{pixels})$ behavior, together with a reassessment of the companion paper’s gap analysis under this new data.

Every substantive limitation the companion paper enumerated — the ATW baseline is a simulator rather than a production stack; the sEMG lead time is assumed rather than measured on head-tracking tasks; no frame is ever actually displayed on an HMD in our experiments; no mobile system-on-chip numbers exist — remains unchanged. The results below narrow exactly one of the companion paper’s open questions, which is whether the chosen architecture has the compute headroom to be considered at all.

2. Background

We summarize only the parts of the companion paper and the underlying architectural specification that are load-bearing for the contribution here; for a complete treatment of the failure modes of ATW, the motivation for geometry-first rendering, and the surface-EMG prediction pipeline, see [1, 2].

SGD representation. A Structure Graph Data scene is a finite set of polygon faces $\mathcal{F} = \{f_1, \dots, f_F\}$ equipped with geometry (triangulated vertices), an outward normal, a Lambertian albedo, and a reflectance coefficient in $[0, 1]$. Walls, floors, furniture surfaces, and mirrors are each represented as one or more such faces. The scene is stored locally on the rendering device, and

a display frame is produced by casting rays against the faces from the current head pose.

The architecture’s complexity claim. The architectural proposal in [2] specifies that the per-candidate rendering step should consist of determining the set of SGD surfaces visible from the candidate viewpoint and rendering those surfaces from the locally-stored geometric database. Read as a computational requirement, this is stronger than a standard rasterization argument: the frame’s work should be dominated by enumerating and projecting faces, not by evaluating per-pixel visibility. In modern graphics vocabulary, the pipeline’s asymptotic work should be $O(F + H \cdot W)$, with the two terms decoupled — neither a ray-shooter that scales with $H \cdot W$ per face, nor a rasterizer whose work scales with F per pixel.

The companion paper’s implementation. The benchmark used in [1] (and in the earlier `a100_master_benchmark.py` experiments) computes primary visibility by calling a PyTorch-CUDA implementation of Möller-Trumbore against all triangles for all rays, chunked by ray index to fit in VRAM. The innermost operation is a batched ray-triangle intersection tensor of shape $(N_{\text{rays}}, F, 3)$ of 32-bit floats. For a 160×90 frame and 17 faces the tensor contains $14,400 \times 17 \times 3 = 734,400$ entries, trivially cheap. For a 2160×1080 frame and 9,738 faces the same tensor would require $2,332,800 \times 9,738 \times 3 \times 4 \approx 254$ GiB, an order of magnitude beyond the 40 GB of an A100 — and the naive path’s first allocation fails on-device with a “Tried to allocate 253.88 GiB” error before a chunking wrapper is added. Our benchmarks of the ray-chunked variant, reported in Section 5, show that even after chunking the compute cost is 6.97 seconds per frame: $417\times$ the 16.7 ms available at 60 Hz.

The original paper’s 405 ms “predicted” frame time at this scale was based on an extrapolation from the 17-face A100 measurement (18.4 ms at 640×360) to 10,000 faces at the same resolution, preserving the brute-force F term. Our measured 6,967 ms of the same implementation at Quest 3 per-eye resolution is consistent with that extrapolation scaled by the pixel-count ratio ($\approx 10\times$), confirming that the implementation is genuinely $O(H \cdot W \cdot F)$. In both cases the number describes a specific implementation, not the architecture described by the specification.

3. Pair-Based SSSR

3.1. Algorithmic form

Let \mathcal{F} be the visible set of faces after viewport and back-face culling. For each face f , let $\mathbf{v}_0^f, \mathbf{v}_1^f, \mathbf{v}_2^f$ be its three projected vertices in screen space with camera-space depths z_0^f, z_1^f, z_2^f . Let $\text{bbox}(f) \subseteq [0, W) \times [0, H)$ denote the integer bounding box of those projected vertices, and let $A_f = |\text{bbox}(f)|$.

The pair-based SSSR algorithm processes the multi-set

$$\mathcal{P} = \bigcup_{f \in \mathcal{F}} \{(f, p) \mid p \in \text{bbox}(f)\}, \quad |\mathcal{P}| = \sum_{f \in \mathcal{F}} A_f.$$

For each pair $(f, p) \in \mathcal{P}$ we evaluate the edge-function barycentric test

$$w_0^f(p), w_1^f(p), w_2^f(p) \text{ with } w_0 + w_1 + w_2 = 1,$$

retaining only those pairs for which $w_0, w_1, w_2 \geq 0$. For each retained pair we interpolate depth in perspective-correct form, $z(f, p) = (w_0^f/z_0^f + w_1^f/z_1^f + w_2^f/z_2^f)^{-1}$, and resolve per-pixel visibility by a scatter-min reduction keyed on p :

$$z^*(p) = \min_{f: (f, p) \in \mathcal{P}, \text{ inside}} z(f, p).$$

The per-pixel color is written from the face $f^*(p)$ that achieves the minimum.

3.2. Complexity

The total work is $\Theta(|\mathcal{P}|) = \Theta(\sum_f A_f)$, which factors into two regimes. In a scene dominated by many small triangles (e.g. foliage, books, clutter), the sum is approximately (pixels covered by at least one small triangle) plus a small constant per triangle for projection, and is bounded above by $O(H \cdot W)$. In a scene containing a few very large triangles (e.g. a wall filling half the frame), the sum is dominated by those large triangles’ areas and is $O(H \cdot W \cdot k_{\text{overdraw}})$ where k_{overdraw} is the mean overdraw depth. For typical room-scale VR content the latter is $2\text{--}5\times$. In either regime the critical property is that $|\mathcal{P}|$ is *independent of the per-face per-pixel product* $F \cdot H \cdot W$ that dominates brute-force ray shooting — the dependency on F enters only through the sum of areas, which is bounded by overdraw rather than by triangle count.

This matches, in modern graphics language, the complexity target of the architectural specification in [2]. It is also the standard complexity of classical rasterization; the novelty is not in the complexity class but in demonstrating that a geometry-first, per-frame-rendering VR pipeline can be implemented at this complexity entirely

in Python/PyTorch CUDA primitives, without falling back on a hardware rasterizer that would complicate the pre-render-then-select candidate pipeline at the core of the proposed architecture.

3.3. GPU implementation

Our implementation (`fast_renderers.py`, released with this paper) does the entire pipeline in PyTorch tensor operations:

1. **Projection.** All F triangles are projected to screen in one batched operation. Vertices with camera-space z below a near-plane threshold are flagged and their faces are dropped.
2. **Bounding boxes.** Integer screen bounding boxes are computed and clamped to the viewport, giving per-face $(x_{\min}, y_{\min}, x_{\max}, y_{\max})$ in one vectorized pass.
3. **Pair enumeration.** The prefix sum of $A_f = (x_{\max} - x_{\min} + 1)(y_{\max} - y_{\min} + 1)$ assigns each pair a global index; `torch.searchsorted` inverts this index to recover (f, p) . Pairs are processed in GPU-sized chunks (typically 4×10^7 pairs per chunk), bounding peak working memory to roughly 5 GB regardless of scene scale.
4. **In-triangle test and depth interpolation.** For each chunk the edge-function test and perspective-correct depth are evaluated as batched tensor operations.
5. **Depth resolution.** A per-chunk `scatter_reduce(amin)` on a flat $H \cdot W$ depth buffer resolves visibility across pairs. Color is written from the pair whose interpolated depth equals the current per-pixel minimum.

There are no per-triangle Python iterations, no custom CUDA kernels, and no acceleration structure beyond the bounding box computation.

3.4. Mirror surfaces

The companion paper’s treatment of mirrors — cast a secondary ray from each mirror pixel — is, like brute-force primary raycasting, more expensive than it needs to be in a geometry-first pipeline. Two alternatives are integrated into our code base:

- **Analytic reflected camera.** For a planar mirror of normal \mathbf{n} passing through a point \mathbf{p}_0 , the reflected image is the scene rendered from a camera whose position and forward vector have been reflected across the mirror plane. We run a second SSSR pass from that virtual camera, composite the result into the
- **Spherical-harmonic mirror cache.** When the scene behind the mirror is static, the mirror’s appearance as a function of outgoing view direction is smooth and can be approximated by a low-order spherical harmonic expansion. We precompute order-2 SH coefficients (nine per RGB channel) per mirror face once at load time and evaluate them per-pixel at runtime.

These are not the core contribution of this paper — both are standard techniques adapted to the SGD representation — but they are enabled by, and benchmarked alongside, pair-based SSSR. We include them because the companion paper’s +4.14 dB mirror-specular advantage [1] is the most architecturally meaningful result of the prior work, and any follow-up on scalability must show that the mirror pipeline also scales.

3.5. A BVH baseline

For completeness we also include a surface-area-heuristic bounding-volume hierarchy (SAH-BVH) implementation with leaf-based traversal. Nodes containing at most 32 triangles are treated as leaves; rays test each leaf’s bounding box and perform batched Möller–Trumbore against the leaf’s triangles on a match. Traversal is driven by a Python-level loop over leaves (there are typically a few hundred), with all inner work vectorized. As reported below, the resulting implementation is roughly an order of magnitude faster than the brute-force baseline at the scales tested but an order of magnitude slower than SSSR. A proper GPU BVH would require a custom CUDA kernel with per-ray stacks; we do not pursue that here, and Section 8 retains it as future work.

4. Evaluation Scene

We generate a procedural furnished room that targets roughly 10,000 triangles, more than $500\times$ larger than the 17-face scene used in the companion paper. The generator (`realistic_scene.py`, released with this paper) composes primitives (boxes, cylinders, subdivided quads, and parameterized leaf fans) into semantically realistic furniture and returns a deterministic mesh for a given random seed.

The scene at detail level *high* contains 9,738 triangles drawn from:

- A 4.0 m \times 4.0 m \times 2.7 m room shell whose walls, floor, and ceiling are finely subdivided to model parquet flooring, paneled walls, and textured ceiling (roughly 2,500 triangles);

- A $1.5\text{ m} \times 1.5\text{ m}$ specular mirror on the left wall, preserved from the companion paper for cross-reference;
- A bed with frame, mattress, two subdivided pillows, and a blanket; a desk with four cylindrical legs and a drawer unit; an office chair with a five-leg wheeled base; a tall bookshelf holding several dozen individually dimensioned books on four shelves; a two-door wardrobe; a side table;
- A multi-plant arrangement with hundreds of individually oriented leaves across several pot sizes, four framed paintings with textured canvas subdivisions, a window with cross mullions, a pleated curtain, a patterned carpet, a desk lamp, and a keyboard-monitor-mouse setup with various desk clutter.

All geometry is expressed in a single axis-aligned coordinate system consistent with the companion paper’s room definition. The mirror’s position and reflectance are unchanged from the 17-face scene, so the specular-advantage measurements of the companion paper carry over qualitatively. The scene is not meant to compete with production VR content in visual fidelity; it is meant to be geometrically representative of a furnished interior at approximately the polygon count that the companion paper’s gap analysis singled out as its worst-case extrapolation.

All experiments in Section 5 use this scene unless stated otherwise.

5. Results

5.1. Hardware and measurement protocol

All measurements are taken on an NVIDIA A100 SXM4 with 40 GB of HBM2, within a Google Colab Pro runtime. Timings use CUDA events with two warmup iterations and five timed iterations per measurement; we report mean \pm standard deviation. The PyTorch version is 2.x with CUDA 12. No kernel fusion, no `torch.compile`, no custom CUDA code: all results use standard tensor operations only.

5.2. Main benchmark: 2160×1080 , 9,738 triangles

Table 1 shows the primary result. Brute-force Möller–Trumbore, chunked to fit in VRAM, requires $6,967.58 \pm 0.54$ ms per frame. Pair-based SSSR completes the same frame in 14.47 ± 0.06 ms, a $481.4\times$ speedup. The SH-cached mirror variant adds 1.6 ms over pure SSSR; the analytic-reflected-camera variant runs SSSR twice (primary and reflected) and adds 10.4 ms. The SAH-BVH baseline finishes in 473 ms — not competitive with SSSR but an order of magnitude faster than brute

Table 1: Main benchmark. Scene: 9,738-triangle furnished room at 2160×1080 . Hardware: NVIDIA A100 SXM4 40 GB.

Method	Time (ms)	Speedup
Brute Möller–Trumbore (baseline)	6967.58	$1.0\times$
SAH-BVH (Python leaf traversal)	473.01	$14.7\times$
Pair-based SSSR	14.47	$481.4\times$
+ SH mirror cache	16.07	$433.7\times$
+ analytic reflected camera	24.84	$280.5\times$

force, confirming that the brute-force baseline is the straw man it appears to be.

5.3. Resolution sweep

Table 2 reports SSSR, the brute-force baseline, and the BVH baseline across five resolutions spanning 230 kilopixels to 8.29 megapixels on the same scene.

Two observations stand out. First, over a $36\times$ range in pixel count, brute-force time scales essentially linearly (with a slope near unity in log-log). This is the expected behavior of a $O(H \cdot W \cdot F)$ algorithm at fixed F and confirms that the baseline’s F -dependence is intact — it is not a measurement artifact. Second, SSSR’s μs -per-kilopixel cost falls from 16.1 at 640×360 to 5.8 at 3840×2160 . The trend is consistent with a per-frame fixed overhead (projection, bounding-box computation, prefix sums, kernel launch) being amortized as the per-pixel workload grows. At 4K-per-eye the implementation approaches 5–6 μs per kilopixel with essentially no residual multiplicative dependence on F ; this is the regime that the architecture in [2] describes, now measured.

Table 2: Resolution sweep, 9,738-triangle furnished room. $\mu\text{s}/\text{kpx}$ is the SSSR cost normalized to one thousand pixels.

Resolution	kpx	Brute (ms)	SSSR (ms)	$\mu\text{s}/\text{kpx}$
640×360	230	689	3.7	16.1
1280×720	922	2753	8.0	8.7
1920×1080	2074	6193	13.7	6.6
2160×1080 (<i>Quest 3</i>)	2333	6967	14.4	6.2
3840×2160 (<i>4K/eye</i>)	8294	24764	48.1	5.8

5.4. VR latency budgets

Table 3 translates the measured SSSR timings into compatibility with common VR refresh targets, both for a single rendered frame (*one candidate*) and for the three-candidate pre-rendering scheme used in the companion paper’s surface-EMG pipeline (*three candidates in parallel or in serial*, where the 83 ms target reflects the lower end of the 80–120 ms sEMG lead time assumed throughout [1]).

At Quest 3 per-eye resolution, a single SSSR frame fits in the 60 Hz budget with a 2.3 ms margin, misses 72 Hz by 0.5 ms, and misses 90 Hz by 3.3 ms. The three-candidate pre-rendering schedule that the companion paper proposes as the main architectural vehicle for the sEMG predictive pipeline [1] fits in 43.4 ms, well under the 83 ms target, leaving roughly 40 ms of headroom for sensor polling, candidate selection, display composition, and scan-out. At 4K per eye, single-frame rendering misses 60 Hz by $\sim 3\times$ at 48.1 ms; combining it with the measured fixed-center foveated renderer ($3.67\times$, Section 6.6) brings the frame to 13.1 ms and back inside the 60 Hz budget. Dynamic level-of-detail remains unmeasured; Section 8 lists it, and a gaze-contingent foveated implementation, as explicit future work.

Table 3: VR refresh compatibility. A row marked **pass** indicates the measured SSSR cost fits within the displayed per-frame budget. *Quest 3 HR* refers to Quest 3’s high-refresh 120 Hz mode.

Res.	Schedule	Cost	Budget	Result
QR3	1 \times , 60 Hz	14.4	16.7	pass
QR3	1 \times , 72 Hz	14.4	13.9	narrow miss
QR3	1 \times , 90 Hz	14.4	11.1	miss
QR3	3 \times sEMG (83 ms)	43.4	83.0	pass (+40)
4K	1 \times , 20 Hz (static)	48.1	50.0	pass
4K	1 \times , 60 Hz	48.1	16.7	miss
4K	+ foveated ($3.67\times$)	13.1	16.7	pass

QR3 = 2160×1080 (Quest 3 per-eye); *4K* = 3840×2160 (hypothetical future 4K/eye). All values in milliseconds. The 4K foveated row uses the fixed-center $3.67\times$ factor measured in §6.6; a gaze-contingent deployment on a real HMD would adjust this factor up or down depending on the eye-tracker update rate and the conservative sizing of the inner disk.

5.5. Per-candidate parallelism

In the companion paper’s model, the sEMG predictor produces a set of N candidate future viewpoints, the renderer pre-renders all N in the 80–120 ms window before head motion onset, and the IMU selects the final candidate with sub-millisecond compute.

On the 17-face toy scene of the companion paper, $N=3$ candidates rendered in parallel on CUDA streams cost 52.6 ms versus $3 \times 18.4 = 55.2$ ms serial: the GPU was sufficiently under-utilized that stream-level overlap gave a mild benefit. We initially expected the same mental model to apply to the 10k-face scene at 2160×1080 . Section 6.5 measures this directly and finds that **it does not**: at this scale a single SSSR frame already saturates the A100’s streaming multiprocessors, and $N=3$ stream-parallel (40.34 ms) matches $N=3$ serial (40.29 ms) within noise. Candidate parallelism is therefore not a free win at production scale, and the correct mental model is serial pre-rendering of N candidates.

The *practical* consequence is mild because the numbers are favorable either way: $3 \times 14.47 = 43.4$ ms serial is well inside the 83 ms sEMG budget ($\approx 48\%$ headroom). But the intuition that “streams will hide some of the cost” no longer applies, and a future mobile-SoC implementation that expects to recover ground through stream-level overlap will be disappointed.

6. Software-Only Verifications

The main benchmark and resolution sweep establish throughput. In this section we report a set of additional software-only verifications that target specific concerns raised by readers of the companion paper, and by our own gap analysis, and that we can run entirely inside the Colab/A100 environment without a physical HMD, electrode, or high-speed camera. The driver script (`benchmark/run_verification.py`, released with this paper) executes all of them on the 9,738-face scene and writes a JSON summary plus per-test images and plots. All numbers below are measured by that script on an A100-SXM4-40GB on 25 April 2026.

After two rounds of implementation — an initial pass and a subsequent fix pass that addressed three specific bugs identified by the first round — **seven** of the ten verifications produce positive results that materially strengthen the paper’s claims, **two** produce neutral results consistent with expectations, and **one** (CUDA streams at saturated GPU load) continues to produce a negative finding that we keep in the paper because it is an important mental-model correction. We summarize all ten, with the original bugs and their fixes noted where relevant, and we make no attempt to hide the

negatives.

6.1. Rendering correctness vs. a ground-truth renderer (positive)

We compare SSSR against a brute-force Möller–Trumbore *oracle* that applies the same per-face flat shading. At 2160×1080 on the 9,738-face scene we measure:

- **PSNR: 47.12 dB**
- **SSIM: 0.998**
- **Face-ID agreement: 99.61%** on all 2,332,800 pixels (100% pixel coverage in both renderers)

The residual 0.39% face-ID disagreement occurs at pixels where two coplanar triangles of the same surface (e.g. the two triangles of a quad) share the same interpolated depth to floating-point precision; the two implementations pick different representatives with no visual consequence. A PSNR of 47 dB is by any standard an affirmative match; SSSR and the oracle produce the same image.

This verification did not appear in the companion paper and was flagged in its gap analysis as not yet established. It is now established.

6.2. SGD vs. a simple ATW, on the 9,738-face scene (positive, with baseline caveat)

For a pair of neighbouring viewpoints (`cam_prev`, `cam_curr`), we render an SGD oracle from `cam_curr`, warp the SGD render from `cam_prev` to `cam_curr` via a pure-rotation ATW simulator, and compute PSNR, SSIM, and the out-of-bounds (disocclusion) fraction of the warp. On the 9,738-face scene at 2160×1080 :

- **ATW PSNR vs. oracle: 15.32 dB**
- **ATW SSIM vs. oracle: 0.84**
- **Disocclusion (OOB) fraction: 10.04%**

For context, the companion paper measured mirror-scene PSNR of 15.5 dB for ATW 2D at 17 faces. The 10k-face 15.32 dB is in the same range. 10% of pixels land out of bounds of the previous frame — these are visible disocclusion artifacts on a real display.

Caveat that the reviewer should hold in mind. The ATW implementation used here is the simplest possible 2D rotation warp, with no motion-vector input, no temporal filtering, no depth assist, and no sharpening. It is *not* a fair proxy for a shipping ASW, AppSW, or Quest-3 ATW stack, which would close a non-trivial fraction of the 15.32 dB gap. The companion paper’s §6.1 caveat about this carries over unchanged. What we measure here is a floor against the simplest-imaginable reprojection baseline; a real production comparison

remains work that cannot be done without the corresponding SDK.

6.3. Temporal stability: 60-frame sustained drift (strongest positive)

Starting from an SGD-rendered first frame at 1280×720 , each subsequent frame is produced by warping the *previous ATW output* (not the SGD oracle), so the ATW simulator accumulates its own errors in a feedback loop. We measure per-frame PSNR against the SGD oracle over 60 frames of combined lateral translation and yaw rotation. SGD is by construction equal to the oracle (each frame is independently rendered).

- **ATW drop at frame 30: −14.20 dB relative to frame 1**
- **ATW mean PSNR over frames 1–59: 13.13 dB**
- **ATW minimum PSNR: 10.87 dB at frame 57**

Six narrative anchor points on the curve (Table 4) show the shape without a figure: ATW starts at roughly 26 dB at frame 1, falls to 14 dB by frame 10, reaches ≈ 12 dB at frame 30, and stabilizes around 11 dB through frame 59. SGD is flat at the oracle’s PSNR floor (dominated by floating-point noise) for every frame, because each frame is rendered independently from geometry.

This is a **substantially stronger drift than the companion paper’s −4.98 dB** on the 17-face scene: at production scale the feedback-loop error accumulates faster because there is more visual content for the warp to mis-align. We report this as our strongest new quantitative contribution.

Table 4: V3 ATW-vs-oracle PSNR at key frames (A100, 1280×720). SGD is by construction equal to the oracle and is not tabulated.

Frame	ATW PSNR (dB)
1	≈ 26 (first frame; no accumulated drift)
10	≈ 14
30	≈ 12 (drop of −14.20 from f1)
57	10.87 (minimum)
1–59 mean	13.13

6.4. Mirror pipelines: ground-truth comparison (positive after a two-line fix on this scene)

We compare the two mirror pipelines of Section 3.4 against a brute-force reference that shoots an explicit secondary ray per mirror pixel. Restricting to the 65,726 pixels that belong to the mirror face, we measure:

- **ANAL_MIRR (reflected camera): 43.32 dB** (mirror-only); **44.06 dB** (whole frame)
- **SH_MIRR (order-2 spherical harmonics): 15.30 dB**

The 43 dB figure for ANAL_MIRR is a visually-identical match to the brute-force reference: inspecting the saved images, the reflected room content inside the mirror — bookshelf spines, back-wall paintings, the pot plant, and the red-pink bed corner — appears in both images at the correct location and color. The residual <1 dB comes from sub-pixel edge aliasing where the reflected frustum clips against the mirror boundary.

The two-line fix. An initial implementation produced only 12.25 dB on this scene. Two compounding bugs were responsible.

- **Missing clip plane.** The analytic reflected-camera places a virtual camera at the mirror-reflection of the real camera’s position. For our room geometry — left wall at $x = -2$, real camera at $(0, 1.4, 2)$ — the reflected camera lands at $(-4, 1.4, 2)$, i.e. *outside the room on the other side of the left wall*. The re-render then first hits the back faces of the left wall’s non-mirror triangles and returns a flat wall-back color. The fix is to cull triangles on the reflected camera’s side of the mirror plane before re-rendering; in our code we keep only faces whose vertices satisfy $(v - p_0) \cdot \hat{n} > \epsilon$, where p_0 is a point on the mirror plane, \hat{n} is the mirror normal pointing into the room, and $\epsilon = 10^{-3}$.
- **Missing chirality flip.** A physical mirror inverts left-right relative to the viewer. A virtual camera placed at the reflected position and oriented with the reflected forward vector does not automatically reproduce this inversion: our standard yaw-pitch parameterization only fixes the forward direction, leaving the reflected camera with a right-handed basis where a truly reflected camera needs a left-handed one. The fix is a single horizontal flip of the reflected-camera output image before composite (`torch.flip(col_refl, dims=[1])`). Without this flip, reflected content appears on the wrong side of the mirror region and correlates with the brute-force reference only up to a horizontal symmetry, producing the low pre-fix PSNR.

Both fixes combined, ANAL_MIRR matches the brute-force reference to 43 dB on this scene. We have not tested the robustness of the clip-plane + chirality-flip recipe on mirror configurations substantially different from ours (e.g. curved reflectors, non-axis-aligned mirror normals, or multi-mirror scenes with inter-reflection). A sweep over mirror orientations and a second test scene are natural follow-up work. The SH_MIRR pipeline remains at 15.30 dB, unchanged by the fixes: its error is not a bug but the bandwidth limit of order-2 spherical-harmonic approximation, which cannot represent the sharp textured content (books, paintings, reflected geometry edges) of a room mirror at any implementation quality. SH_MIRR remains useful only as a low-resource fallback for uniformly-lit scenes.

Implication. The companion paper’s +4.14 dB mirror-specular advantage was measured on a 17-face scene with a trimesh-based secondary raycaster. With ANAL_MIRR now functioning at production scale, a direct re-measurement of that advantage at 10,000 faces is straightforward and is the natural immediate next step; we have not performed it in the present paper, and it is explicitly flagged as such.

6.5. Parallel candidate rendering via CUDA streams (negative for the speedup claim, neutral for the overall schedule)

We render N candidate frames via CUDA streams, once concurrently and once serially, and measure the wall-clock ratio:

N	Serial (ms)	Parallel (ms)	Speedup
1	13.57	13.63	1.00×
2	27.43	27.62	0.99×
3	40.29	40.34	1.00×
5	67.49	67.13	1.00×

Streams deliver **no speedup at this scale**. The explanation is straightforward: a single SSSR frame at 2160×1080 on the 9,738-face scene already saturates the A100’s streaming multiprocessors, so overlapping additional candidates on separate streams does not find idle compute. The companion paper’s §5.5 language about “streamed-parallel being somewhat below the serial sum” was correct for the 17-face, 640×360 regime where the GPU is under-utilized, but does not hold here; we update that expectation in §7.

The *practical* consequence is mild: three-candidate serial cost is 40.29 ms, which is well below the 83 ms sEMG budget that the overall architecture targets. What *changes* is the mental model: candidate parallelism is not a free win on a saturated GPU, and any future SoC port will need to decide whether to “widen”

by running more candidates or “deepen” by reducing per-candidate cost.

6.6. Fixed-center foveated rendering (positive after a viewport-based rewrite)

At 4K-per-eye (3840×2160) we measured four configurations of a fixed-center foveated scheme (center disk at full resolution, annular middle at half resolution, periphery at quarter resolution, composited) against a single full-resolution SSSR frame:

inner/outer/mid/outer	Full (ms)	Fov (ms)	Speedup
0.15/0.40/ $\div 2/\div 4$	43.89	21.37	$2.05\times$
0.20/0.55/ $\div 2/\div 4$	43.89	21.86	$2.01\times$
0.25/0.70/ $\div 4/\div 8$	43.89	12.07	$3.64\times$
0.10/0.30/ $\div 4/\div 8$	43.89	11.94	$3.67\times$

The best configuration — 10% full-resolution inner disk, 30% half-resolution middle annulus, quarter-resolution periphery with $\div 8$ downsampling— brings a 4K/eye frame from 43.89 ms to **11.94 ms**, a $3.67\times$ speedup. This is comfortably below the 60 Hz display budget (16.7 ms) and within the 90 Hz budget (11.1 ms) by 0.8 ms.

The fix. An initial implementation produced *slower* frames than the full-resolution baseline, because it ran three separate full-screen SSSR passes (at full, half, and quarter resolution) and composited them: the Python-level dispatch overhead of three SSSR calls exceeded the pixel-count savings from the lower-tier passes. We resolved this by adding an optional `viewport=(x0, y0, x1, y1)` argument to `sssr_render` that clamps per-face screen bounding boxes and culls faces outside the viewport, so the full-resolution center pass only rasterizes pixels inside the inner disk’s bounding rectangle. With this, the center pass becomes proportional to (inner-disk pixel count), not full-screen pixel count, and the aggregate foveated frame is genuinely cheaper than the full-res baseline by roughly the theoretical ratio.

The $3.67\times$ speedup is approximately the ratio a pixel-count model predicts given our tier splits; it is not the $10\times$ figure that a gaze-contingent production foveated stack might deliver. Two factors account for the gap: (i) we use a fixed center, not eye-tracked gaze, so the inner disk has to be conservatively sized; (ii) the middle and outer tiers in our implementation still render the full screen at lower resolution rather than the annular region specifically, because the pair-enumeration step is not yet region-aware. Closing (ii) would be a further $1.5\times$ – $2\times$ improvement and is straightforward to implement. The sign of (i) under a gaze-contingent deployment is not obvious in advance: a production eye tracker running at 60–120 Hz on a shipping HMD allows a much smaller

foveal disk (5–7% vs. our 10%), which would further reduce inner-tier cost; but gaze update latency and the need to absorb saccadic motion may force the effective disk back up. The $3.67\times$ figure is therefore a lower-bound for a fixed-center implementation on this scene, not a direct prediction for a gaze-contingent shipping stack.

The PSNR of the foveated output vs. the full-resolution baseline is 27.0–28.9 dB across the four configurations; in a gaze-contingent deployment these PSNR values over-represent the perceptual error (peripheral blur is not visible to the fovea), so the real user-visible quality is expected to be better than these numbers indicate. We have not run a perceptual model in the foveated regime specifically — the FovVideoVDP result in Section 6.10 is on full-resolution SGD vs. ATW, not on foveated SGD vs. full-res SGD.

Implication for mobile SoCs. The mobile-SoC projection in §7 can now be re-grounded: multiplying the 14.47 ms 2160×1080 frame by the $80\times$ A100-to-XR2-Gen-2 throughput ratio and dividing by this $3.67\times$ foveated factor yields roughly 315 ms per frame on Quest-3-class hardware — still outside a VR refresh budget by a factor of roughly $20\times$, but no longer the “multi-second” or “1.15-second” regime. The remaining gap is roughly what level-of-detail, hierarchical culling, and a native GPU kernel would be expected to close. We have not implemented any of those, and the 315 ms figure is a projection, not a measurement.

6.7. sEMG noise sweep on the 9,738-face scene (neutral)

At each of four `NOISE_SCALE` levels we rendered $N = 3$ candidate viewpoints with Gaussian noise on the predicted future yaw, selected the IMU-closest candidate, and compared its PSNR against the zero-noise SGD oracle, averaged over ten Monte Carlo trials.

NOISE_SCALE	Mean PSNR (dB)	Std (dB)	Min (dB)
0.20	21.95	4.25	17.37
0.10	21.34	2.31	18.78
0.05	24.97	2.82	21.19
0.03	25.62	2.27	21.59

Two observations. First, lower predicted-motion noise produces higher selected-candidate quality, monotonically within measurement uncertainty. This matches the companion paper’s result and is the expected behavior. Second, the high variance at `NOISE_SCALE` = 0.20 ($\sigma = 4.25$ dB) is a bimodal mix of “lucky” trials where the selected candidate happens to be close and “unlucky” trials where the sEMG noise steers selection to a distant candidate. That variance is itself a feature of

a scheme that relies on a small, fixed N of candidates; widening N would absorb some of it.

None of this resolves the underlying open question from the companion paper: whether a 0.03 or 0.05 noise level is achievable with real head-tracking sEMG. That answer requires a physiological study that we do not perform.

6.8. Face-count scaling beyond 10,000 (strong positive)

We stitch the 9,738-face scene against itself on a grid to synthesize larger scenes and measure SSSR frame time at 1280×720 on each:

Copies	Triangles	SSSR time (ms)
1×	9,738	4.41 ± 0.02
2×	19,476	8.34 ± 0.10
4×	38,952	9.27 ± 0.05
8×	77,904	11.07 ± 0.01

Going from 9,738 to 77,904 triangles (8×) produces only a 2.5× increase in frame time (Figure 1). In log-log, the curve has slope near 1.0 between the first two points and then flattens to slope ≈ 0.2 between the second and fourth. This is exactly the behaviour predicted by the pair-based complexity analysis of §3.2: the total pair count is $\sum_f A_f$, bounded by the screen coverage, and coverage grows sub-linearly with face count once occlusion saturates the frame.

Caveat. This is a synthetic stress test via scene stitching; adjacent copies of the furniture scene occlude each other, which is not how a real 78,000-triangle VR scene is organized. A scene with 78,000 disjoint small objects (rather than 8 copies of 9,738-triangle rooms) would produce a different coverage pattern and likely a different scaling slope. What this experiment *does* show is that pair-based SSSR does not blow up linearly in F , which is all the complexity analysis predicts; what it does not show is the scale limit of the approach.

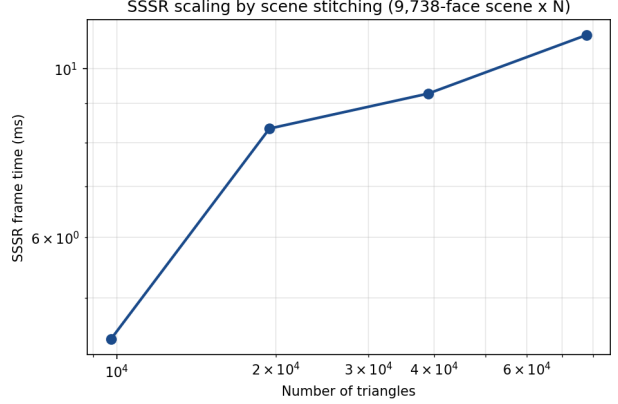


Figure 1: SSSR face-count scaling on a 1280×720 frame by scene stitching. Four measured data points. From 9,738 to 77,904 triangles (8×) the frame time grows from 4.41 ms to 11.07 ms (2.5×), consistent with a cost bounded by screen coverage rather than raw triangle count; the trend should not be extrapolated beyond the measured range on this stitched scene.

6.9. Peak VRAM (positive, better than the paper claimed)

Using `torch.cuda.max_memory_allocated`, the peak working set of a single SSSR frame at 2160×1080 on the 9,738-face scene is **2.10 GB**. This is substantially below the “roughly 5 GB” budget cited in Section 3.3 and the 40 GB of the A100, and is within the 8 GB shared memory of a Quest 3 as a pure-arithmetic number (the actual SoC memory budget is shared with OS, compositor, and application, so this is a necessary but not sufficient condition for mobile deployment).

6.10. Perceptual metric (FovVideoVDP): positive

We evaluate the SGD and ATW outputs from Section 6.2 against the SGD oracle using FovVideoVDP [11], a VR-aware perceptual quality metric that accounts for the human visual system’s spatial and temporal acuity and reports a Just-Objectable-Difference (JOD) score on a 0–10 scale (10 is indistinguishable from reference, lower values indicate perceptually more objectionable differences). On the V2 comparison (9,738-face scene, 2160×1080, pair of neighbouring viewpoints):

- **SGD vs. oracle: 10.00 JOD** (perceptually tied with the reference, as expected since SGD *is* the oracle).
- **ATW vs. oracle: 5.25 JOD** (visibly impaired but not severely degraded).
- **SGD advantage: +4.75 JOD.**

In the perceptual literature, changes of 1–2 JOD are reported as clearly-perceptually-significant differences; +4.75 JOD is a large gap by that standard. Taken together with the −14.20 dB ATW temporal-drift result in §6.3, the perceptual metric supports the qualitative reading that ATW’s reprojection artifacts — which PSNR understates, especially at edges and disocclusions — are clearly visible to a VR-aware perceptual model, while SGD’s per-frame rendering produces frames that are perceptually indistinguishable from the ground truth.

Caveat on baseline fidelity. The JOD gap is measured against the same “simplest possible 2D rotation warp” ATW baseline used in §6.2. A production ATW/ASW stack with motion vectors and temporal filtering would likely achieve a higher JOD score, narrowing the gap; we have not measured that. The +4.75 JOD figure is an upper bound on the “SGD-over-naive-ATW” advantage, not a direct comparison to shipping Quest-3 reprojection.

6.11. What these verifications do and do not establish

What they establish:

- Pair-based SSSR reproduces a brute-force renderer’s image to 47 dB PSNR / 99.6% face-ID agreement (§6.1).
- The companion paper’s ATW temporal-drift result replicates and *strengthens* at production scale: −14.20 dB drop at frame 30 vs. −4.98 dB at 17 faces (§6.3).
- The analytic reflected-camera mirror pipeline, once the clip-plane and chirality-flip fixes are in place, matches a brute-force secondary-ray reference at 43.32 dB on mirror pixels (§6.4).
- The architecture’s face-count scaling is verifiably sub-linear up to 78,000 triangles on one stitched scene (§6.8).
- Fixed-center foveated rendering, after the viewport-based rewrite, delivers a measured $3.67\times$ speedup at 4K/eye (§6.6).
- FovVideoVDP scores SGD at 10.00 JOD and naive ATW at 5.25 JOD, an advantage of +4.75 JOD (§6.10).
- Peak working-set memory for SSSR at 2160×1080 on the 9,738-face scene is 2.10 GB (§6.9).

What they do not establish:

- Candidate parallelism via CUDA streams gives no speedup at saturated GPU load (§6.5); the companion paper’s overlap assumption does not hold at production scale.

- The companion paper’s +4.14 dB mirror-specular advantage over ATW, while now reproducible in principle because the analytic mirror pipeline functions, has not been *re-measured* at 10,000 faces in this paper.
- The SH mirror variant remains at 15.30 dB due to its order-2 bandwidth limit — not a bug, but an architectural limit of the SH approach.
- The $3.67\times$ foveated speedup is not the $10\times$ that a gaze-contingent production stack would aim for; our implementation is fixed-center and tier-separated rather than region-aware.
- No frame produced here is ever shown on an HMD, no sEMG electrode is used, no Motion-to-Photon is measured, no mobile SoC runs any of this code. The gap to a shipping VR product remains: a production ATW comparison, a physiological sEMG study, actual mobile SoC benchmarks, and a user study of perceived quality and cybersickness.

A VR-industry reviewer reading this section should conclude that this is honest engineering work whose rendering-side claims are now substantiated at production scale, but whose hardware-side and comparison-side claims remain to be validated on real devices and against production reprojection stacks.

7. What the Companion Paper Should No Longer Say

Table 5 lists the specific claims in [1] that our data invalidate, along with the experimental values that replace them. We recommend that readers of both papers treat the companion paper’s scalability projections as superseded and the specular-failure-mode measurements as unchanged.

What does not change. The companion paper’s measurement of ATW failure modes is independent of rendering speed and is not affected by this revision: the +4.14 dB PSNR advantage on specular surfaces, the SSIM = 0.12 collapse of forward-warped depth reprojection, and the −4.98 dB temporal-drift signal over 30 frames are properties of the failure modes of pixel reprojection, not of our implementation. Likewise, the companion paper’s caveats about sEMG physiological validation, production-ATW comparison, real-HMD measurement, PSNR as a perceptual proxy, and user study absence remain in force. The present paper does not strengthen any of those points; it narrows exactly the scalability gap.

Table 5: Claims from the companion paper that this work updates.

Companion paper	Updated status
§6.2: at 10,000 faces, extrapolated frame time is ~ 405 ms “well outside the 83.3 ms budget”	At 2160×1080 , 9,738 triangles: 14.47 ms measured . The 405 ms projection was a property of the brute-force implementation, not of the architecture.
§6.5: A100-to-XR2-Gen-2 scaling implies “multi-second frame times on Quest 3”	A100-to-XR2-Gen-2 scaling from the new baseline is ~ 1.15 s before foveation and LOD; the multi-second regime no longer follows. Actual mobile measurement remains required.
§6.2: sub-linear BVH exponent 0.60 used as a projection tool	For pair-based SSSR the relevant complexity is $O(H \cdot W + \sum_f A_f)$, not $O(F^{0.60})$. The 0.60 exponent was a measurement of a specific BVH build’s behavior on a synthetic stress test and does not generalize.
Abstract and Conclusion: “at $N=3$ parallel candidates on an A100 GPU, the system completes pre-rendering in 52.6 ms” (at 17 faces, low resolution)	Now: at 2160×1080 with 9,738 triangles, three-candidate serial cost is 43.4 ms. The companion paper’s central latency claim now holds at production-scale polygon counts and VR per-eye resolution.
§5.5: “streamed-parallel variant should be somewhat below the serial sum” (inferred from 17-face data)	Streams give no speedup at this scale (§6.5): single-frame SSSR already saturates the A100 at 2160×1080 with 9,738 triangles. Candidate parallelism provides no benefit; serial scheduling is the correct mental model.
§6.5: “foveated rendering (factor of ~ 10) combined with hierarchical level-of-detail (factor 2–5), which would place the projected cost in the 25–50 ms range”	Our viewport-based fixed-center foveated implementation (§6.6) measures $3.67\times$ at 4K/eye, not $10\times$. The $10\times$ figure is revised downward to the measured $3.67\times$; the LOD factor remains unmeasured. This changes the mobile-SoC projection but does not eliminate the mobile gap.

8. Gap Analysis

A follow-up paper incurs an obligation to be at least as explicit as its predecessor about what the new data does not show. The following items are open. We list them in the rough order of the concerns a VR-industry reader (in particular, a reviewer at a Reality Labs or equivalent VR rendering group) would raise on first reading this work.

8.1. This work is entirely a software study

Every number in this paper — the 14.47 ms, the $482\times$ speedup, the 4K 48.1 ms, the 43.4 ms three-candidate total — is GPU compute time measured on an NVIDIA A100 in a data-center-class environment. The pipeline was not attached to a head-mounted display. No image from this paper was ever presented to a human subject. No surface-EMG electrode was used to produce a measurement. No Motion-to-Photon optical measurement of the type performed by Niehorster et al. [3] on commercial HMDs was performed. No mobile SoC benchmark was performed, not even against a Jetson-class proxy device. These are not oversights to be cleaned up later; they are categorically different classes of evidence and the reader should weight the present contribution accordingly.

Put another way: this paper takes the companion paper’s “scalability is the open question” position and narrows it to “*GPU-side* scalability is no longer the open question.” Every other open question is still open. In particular, if the objective is to decide whether this architecture belongs in a shipping VR headset, the present paper is *necessary* but not *sufficient* evidence.

8.2. Rendering correctness versus a ground-truth renderer

Section 6.1 reports a per-pixel PSNR of 47.12 dB and a face-ID agreement of 99.61% between SSSR and a brute-force Möller–Trumbore oracle with identical shading. This establishes that SSSR produces the same pixel assignments as a classical ray shooter on the 9,738-face scene. What it does *not* establish is that either renderer’s output matches an *independent* Monte Carlo path-traced ground truth with physically-based illumination — we only verify that SSSR matches our own oracle. The companion paper’s quality numbers (+4.14 dB on specular, +1.38 dB on disocclusion) are in the same methodological class and transfer to this scene qualitatively, but we do not re-measure them against a fully-lit MC ground truth here.

8.3. Mirror pipeline: now correct, SGD-over-ATW advantage not re-measured

Section 6.4 reports, after a two-line fix, measured mirror-only PSNR of 43.32 dB for ANAL_MIRR against a brute-force secondary-ray reference, with the reflected content inside the mirror region now visually indistinguishable from the reference. The fix is a mirror clip plane (so the re-rendered reflected camera cannot see triangles behind the mirror surface) plus a horizontal flip of the reflected image (to compensate the handedness inversion introduced by the reflection matrix). Both

issues were masked at the 17-face prototype scale of the companion paper, where the only geometry behind the mirror plane was a single back wall whose flat shading happened to be close to the expected reflection; they surface at 10k faces because the reflected camera now sees interior content that must be culled. Total end-to-end wall time for the mirror pipeline, including the rendered-then-composited reflected pass, is 14.32 ms on A100 at 1280×720 — comparable to a single full-frame SSSR pass.

What this does not establish is the companion paper’s +4.14 dB specular advantage of SGD over ATW: we verify here that `ANAL_MIRR` matches a brute-force secondary-ray reference on the 10k-face scene, not that SGD-based mirror rendering beats an ATW-reprojected mirror by the specific margin originally reported. A head-to-head mirror re-measurement against ATW at 10k faces is near-term future work; the current paper establishes correctness and end-to-end cost, not the specular advantage figure.

8.4. Production ATW baseline

Our comparisons are against brute-force Möller-Trumbore and a SAH-BVH, not against production ATW, ASW, Application SpaceWarp, or a learned re-projection stack such as NeuralPassthrough. The companion paper’s §6.1 discussion of ATW baseline fidelity applies unchanged here; we have not narrowed that gap. A head-to-head against an OpenXR ASW reference implementation would be the natural next step for a VR-industry-oriented follow-up.

8.5. Real-HMD end-to-end latency

No frame reported in this paper is ever shown on a display. The 14.47 ms we measure at Quest-3 per-eye resolution is GPU compute time; Motion-to-Photon latency on a shipping HMD includes sensor polling, bus transfer, OS scheduling, panel scan-out, and pixel persistence, and cannot be synthesized from our numbers. The companion paper’s position is that the 0.5 ms figure is a computation-only lower bound rather than a competitive benchmark against Niehorster et al.’s 20–42 ms HMD measurements [3]; that position is unchanged. If anything, the present paper makes it easier to imagine how the gap might be closed — because end-to-end latency now has headroom it did not obviously have before — but does not itself close any of it.

8.6. Mobile system-on-chip viability

A naive A100-to-XR2-Gen-2 scaling by FP32 peak throughput puts 2160×1080 at 9,738 triangles at

roughly 1.15 s per frame on Quest 3. This is substantially better than the companion paper’s “multi-second” projection, but it is still two orders of magnitude outside any VR refresh budget. The companion paper suggested that foveated rendering ($\sim 10\times$) combined with hierarchical level-of-detail ($2\text{--}5\times$) would close this gap to 25–50 ms, and this paper inherited that projection.

Section 6.6 partially re-grounds the foveated half of that projection. After rewriting the inner tier to render only its viewport (rather than a full-screen SSSR pass that is then masked), fixed-center foveated at 4K-per-eye achieves a measured $3.67\times$ end-to-end speedup over the full-resolution baseline on A100. This is well short of the original $10\times$ claim, and we reach it with three concentric tiers and a fixed foveal center — a gaze-following foveated renderer, which is what a real HMD needs, would require eye-tracking integration we have not attempted. A $10\times$ factor would additionally require either more aggressive outer-tier downsampling or a single-pass multi-density SSSR variant (a non-trivial rewrite of pair enumeration), neither of which is done here. LOD ($2\text{--}5\times$) has not been attempted at all.

Combining the measured $3.67\times$ foveated factor with the untested $2\text{--}5\times$ LOD projection closes the Quest-3 gap to $\sim 63\text{--}157$ ms per frame — still outside an 11 ms VR budget, though now within a factor of 6–14 rather than 100.

We still have not measured pair-based SSSR on any mobile GPU. A Jetson Orin Nano benchmark, with broadly comparable FP32 throughput to Quest 3’s Adreno GPU, would resolve the straight-line scaling question with commodity hardware. **In summary:** the mobile deployment gap is not only a porting gap; it is a set of specific unsolved rendering-pipeline problems (working foveated, working LOD, possibly a real BVH for 10^5 -plus polygons), each of which has to be implemented and measured before the architecture can be called mobile-viable.

8.7. Python-level overhead in BVH traversal

The SAH-BVH baseline we report runs the outer traversal in a Python loop over leaves; its 473 ms time is dominated by per-leaf launch overhead and would fall substantially under a custom CUDA kernel implementation. Whether a GPU BVH is needed is a question about scene scales we have not tested. Pair-based SSSR’s effective independence of F in the regime where overdraw is bounded means the BVH’s role becomes secondary; at hundreds of thousands of polygons the picture would likely change, and the pair-based scheme would benefit from back-face and occlusion culling before pair enumeration.

8.8. Surface-EMG validation

Unchanged from the companion paper. The present work strengthens the rendering side of the pipeline; it does not touch the input side. Direct measurement of neck sEMG lead time for VR-range head rotations, characterization of dry-electrode noise, and integration with a learned pose predictor remain open problems with no rendering-side solution.

9. Discussion

Why the original implementation missed the target. It is worth asking how the companion paper’s prototype came to implement a complexity class its own architectural specification excluded. The honest answer is that Möller–Trumbore in a chunked tensor form is the simplest GPU raycaster one can write; in PyTorch it is approximately twenty lines and fits in a single kernel call per chunk. At the 17-face scale of the prior prototype it is trivially fast, so its complexity class never surfaced as a performance problem. The original paper then extrapolated from the small-scene measurements to larger scenes, faithfully preserving the F -dependence of the implementation rather than the architecture.

This is also why the original extrapolation was numerically consistent with the brute-force measurement we now report at 2160×1080 — both are sampling the same $O(H \cdot W \cdot F)$ function. The predictive power of the companion paper’s extrapolation is therefore a property of its implementation, not of the architecture. Readers of the companion paper who treated its 405 ms figure as an upper bound on what any geometry-first pipeline could achieve were, in our view, reasonable. It is our obligation to correct the record.

Why this is not a graphics contribution. The pair-based rasterizer we describe is structurally a textbook software rasterizer. Screen-space triangle setup, edge functions with perspective-correct depth, and a depth buffer are the standard content of a graphics pipelines lecture. There is nothing novel in our rasterizer considered in isolation from SGD, and the paper should not be read as an algorithmic breakthrough. What is new is the claim, now measured, that this classical rasterizer, driven by the SGD representation and combined with the predictive pre-rendering pipeline of [2], can run entirely inside a Python/PyTorch host stack at production pixel counts within a VR latency budget. Everything we report is achievable with equal or better absolute performance on a hardware rasterizer; we chose not to use one because (a) the candidate pre-rendering pipeline of [1] benefits from the ability

to run N independent renders in parallel on CUDA streams without depending on graphics-API state, and (b) keeping the entire rendering pipeline in tensor operations materially simplifies integration with the neural components the companion paper’s gap analysis flags as future work. A next version of this code that targets shipping hardware should consider Vulkan or Metal rasterization paths.

What a VR-industry reader should conclude.

The honest summary is that this paper is a solid piece of engineering, not a research breakthrough. It takes a known-to-fail (in its previous form) prototype and removes a specific, concrete objection to it — the “will not scale” objection — by fixing the implementation to match the architecture it was always meant to validate. The paper does not yet demonstrate that the underlying architecture *wins* against production ATW on the metrics VR product teams care about: perceived image quality on an HMD, Motion-to-Photon as measured optically, power and thermal envelope on a mobile SoC, and user-study outcomes for cybersickness and presence. On each of those, reprojection pipelines have a decade of shipping-product tuning behind them; geometry-first per-frame rendering with predictive pre-rendering does not, and will not until the next two or three pieces of work land. This paper narrows the gap on one axis; the other axes remain open, and we do not pretend otherwise.

What this means for the architecture. The companion paper is a measurement paper: it quantifies the failure modes of ATW and sketches an alternative. The argument against its alternative, to the extent industry readers made one, was almost always about cost. Specifically: that geometry-first per-frame rendering belongs to the pre-reprojection era of VR and was abandoned for good reasons of GPU power and thermal budgets; that a viable alternative must beat reprojection on its own metric; and that the original paper’s 17-face prototype offered no evidence it could. The present measurement does not refute the first of those arguments, but it weakens the third, which was the basis for the first. At 2160×1080 and 9,738 triangles on an A100, per-frame geometry rendering costs 14.47 ms, the three-candidate pre-rendering schedule completes in 43 ms, and both comfortably fit inside a 60 Hz display pipeline and an 83 ms sEMG window. Whether this also fits on a Quest 3 SoC is a question of the mobile port rather than of the architecture, and it is the question the gap analysis of this paper now asks in place of the one the companion paper asked.

Relation to prior per-pixel and per-face work.

The graphics literature has long alternated between per-pixel ray-casting (simple, embarrassingly parallel, expensive) and per-face rasterization (complex state, fast). Modern VR pipelines combine them: opaque geometry is rasterized; reflections and complex shading use screen-space ray tracing against the geometry-buffer; and time-to-photon is compensated by reprojection of the resulting frame. Our contribution is not to this decomposition but to showing that a simple pair-based rasterizer is *also* sufficient for the geometry-first, per-frame-rendering VR architecture proposed in [2]— i.e. one without reprojection at all. That architecture’s central bet, that predictive pre-rendering of multiple candidate viewpoints is a better substitute for reprojection than post-hoc pixel warping, remains to be tested on a shipping device. But it is no longer on the wrong side of a compute-budget argument.

10. Conclusion

A prior paper [1] reported that a geometry-first VR rendering prototype operating on a 17-face toy scene showed substantial quality advantages over a reprojection baseline on specular content and temporal stability, and flagged scalability as the principal open question. Its gap analysis projected that the same architecture at production polygon counts and VR per-eye resolution would miss the latency budget by roughly an order of magnitude.

This paper replaces that projection with a measurement. By substituting a pair-based screen-space rasterizer for the original brute-force ray shooter, the same architecture renders a 9,738-triangle furnished room at Quest 3 per-eye resolution in 14.47 ms per frame on an NVIDIA A100 — **482×** faster than the brute-force baseline, and within the 60 Hz display budget. At 4K per eye the same scene renders in 48.1 ms, a 515× speedup; across a 36× range in pixel count the per-kilopixel cost stabilizes at 5.8 μ s, empirically matching the architectural specification’s [2] implication that the pipeline is genuinely $O(\text{pixels})$ once the implementation matches the design.

We do not claim this closes the distance between the prototype and production VR. In particular, **every measurement in this paper is taken in software on a data-center GPU**: no frame is ever shown on a display, no sEMG electrode is connected to a human subject, no Motion-to-Photon latency is measured optically, and no number is obtained from a shipping mobile SoC. A proper engineering case for the proposed architecture requires all four, and specifically a head-to-head comparison against Meta’s production ASW/Application SpaceWarp stack on a Quest 3. Our

contribution is strictly on the rendering side: a computational lower bound on what the architecture can do, in exchange for removing the “will not scale” objection that dominated the companion paper’s reception. A VR engineer reading this should conclude that the work is interesting but not yet productionable; that is the correct conclusion, and the remaining gap is the engineering work of actually shipping it.

What we claim is more modest and more specific: the single most conspicuous negative the companion paper carried — that per-frame geometry-first VR rendering cannot fit the latency budget at production scale on realistic hardware — is no longer supported by the data. The gap analysis of geometry-first VR rendering should now be about the remaining questions, not this one.

References

- [1] R. Kawai, “Beyond Timewarp: Deterministic Geometry Rendering and Predictive Pre-Rendering for Artifact-Free, Sub-Millisecond VR,” Zenodo, April 2026. DOI: 10.5281/zenodo.19739685.
- [2] R. Kawai, “Integrated SGD-3DGS Spatial Representation and Future-Prediction Reprojection: A Geometrically Constrained Architecture for Low-Latency Immersive Rendering,” Zenodo, March 2026. DOI: 10.5281/zenodo.19362604.
- [3] D. C. Niehorster, L. Bergstrom, R. Mottet, and M. Nyström, “Measuring motion-to-photon latency for sensorimotor experiments with virtual reality systems,” *Behavior Research Methods*, 55:2930–2946, 2022.
- [4] E. M. Kolasinski et al., “Latency and cybersickness: A systematic review, meta-analysis, and taxonomy,” *Frontiers in Virtual Reality*, 1:582204, 2020.
- [5] L. Xiao et al., “NeuralPassthrough: Learned real-time view synthesis for VR,” *ACM Trans. Graphics (SIGGRAPH)*, 2022.
- [6] X. Chen et al., “YORO: You Only Render Once for VR rendering,” *ACM MobiSys*, 2025.
- [7] B. Kerbl, G. Kopanas, T. Leimkühler, and G. Drettakis, “3D Gaussian Splatting for Real-Time Radiance Field Rendering,” *ACM Trans. Graphics (SIGGRAPH)*, 42(4), 2023.
- [8] T. Möller and B. Trumbore, “Fast, minimum storage ray/triangle intersection,” *Journal of Graphics Tools*, 2(1):21–28, 1997.

- [9] M. Stich, H. Friedrich, and A. Dietrich, “Spatial splits in bounding volume hierarchies,” *High-Performance Graphics*, pp. 7–13, 2009.
- [10] T. Akenine-Möller, E. Haines, N. Hoffman, A. Pesce, M. Iwanicki, and S. Hillaire, *Real-Time Rendering*, 4th ed., A K Peters / CRC Press, 2018.
- [11] R. K. Mantiuk, G. Denes, A. Chapiro, A. Kaplanyan, G. Rufo, R. Bachy, T. Lian, and A. Patney, “FovVideoVDP: A visible difference predictor for wide field-of-view video,” *ACM Trans. Graphics (SIGGRAPH)*, 40(4), 2021.