

Splatfiction: High-Fidelity 3D Gaussian Splatting

Training in WebGPU

Tomáš Sikora
Reaigen s.r.o.
Slovakia

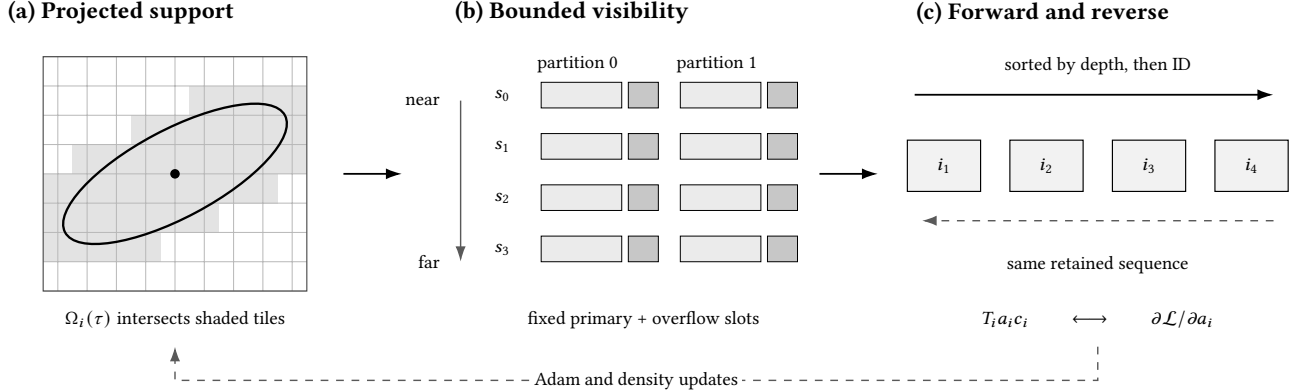


Figure 1: A bounded differentiable splatting loop. Opacity-aware support determines candidate tiles; finite depth partitions retain a subset of contributors; the retained set is ordered for forward compositing and reverse differentiation. The footprint is an analytic illustration, not a reconstruction. Capacity-limited retention remains an approximation.

Abstract

Splatfiction brings high-fidelity 3D Gaussian training into the browser, extending browser workflows beyond viewing an already trained scene. The WebGPU system combines calibrated anisotropic image formation, opacity-aware tile admission, bounded visibility retention, integer-based gradient transport, and state-preserving population refinement. It retains spherical-harmonic appearance through degree three. Three complete Mip-NeRF 360 suites share a recorded commit, deployment, and benchmark-worker hash. At 30,000 iterations and a one-million-Gaussian cap, they achieve 27.70–27.76 dB mean held-out PSNR and 0.799–0.801 SSIM across all nine scenes. The reference suite records 8.06–17.38 minutes of synchronized optimizer time per scene on the RTX 4090 GPU class. Separate Counter experiments associate reducing the maximum SH degree from three to one with a 0.756 dB PSNR loss and show a non-monotonic response to stronger late-stage MSE weighting. These findings inform our representation and objective choices. The Gaussian optimizer needs no server-side fallback, and trained fields can be exported for downstream use; camera calibration remains an input to this stage. The results establish working browser-native training on the tested configurations, with explicit numerical and resource tradeoffs. Bounded retention remains approximate, and a matched native-baseline comparison is outside the present evaluation.

Keywords

3D Gaussian Splatting; WebGPU; differentiable rendering; browser computing; novel-view synthesis.

1 Introduction

Splatfiction moves the creation of a Gaussian scene into the browser, rather than limiting the browser to viewing an already trained asset. It takes photographs with calibrated camera poses, optimizes a three-dimensional Gaussian representation through WebGPU, and produces a field that can be exported for downstream use. The Gaussian optimizer requires no server-side fallback; camera calibration remains an explicit input to this stage.

Three-dimensional Gaussian Splatting (3DGS) models a scene with anisotropic primitives whose position, shape, opacity, and directional appearance are learned from photographs [7]. Bringing that training loop into a browser requires rendering, differentiation, parameter updates, and population refinement to work together within a portable GPU interface. This is a substantially broader capability than browser playback.

Our design preserves the expressive representation while controlling its execution costs. Calibrated cameras and full anisotropic Gaussians are combined with analytic ellipse-tile admission, bounded depth-partitioned retention, and a shared forward/reverse order. Portable reductions feed Gaussian-wise updates. Population refinement carries forward appearance and relevant optimizer history instead of treating every change as a fresh initialization. Figure 1 summarizes the loop.

WebGPU’s execution constraints shape these choices: core WGSL atomics are integer typed, optional features vary by adapter, and storage limits must be negotiated [14, 15].

Visibility buffers, parameters, optimizer state, and training images share a finite memory budget. The optimizer must also coexist with the application’s presentation and interaction work.

We treat camera projection, projected support, visibility order, gradient transport, and appearance capacity as quality-critical parts of the system. A browser port can complete while introducing approximation error or suppressing useful updates. The engineering objective is therefore not completion alone, but useful reconstruction quality within a declared resource budget.

The evaluation follows the same principle. Three complete nine-scene suites establish the benchmark result; separate Counter experiments examine appearance capacity and objective settings. These targeted findings inform our development choices without attributing the total image quality to every subsystem. The supported product capability is local Gaussian training and export on the tested desktop configurations, not universal device coverage or a measured advantage over a matched native trainer.

2 Related Work

Representation and image formation. The original 3DGS method combines anisotropic Gaussians, spherical-harmonic (SH) appearance, differentiable rasterization, and density control [7]. EWA splatting provides the projected elliptical-footprint foundation [20]. Mip-Splatting studies scale-dependent filtering, while StopThePop demonstrates the importance of visibility ordering for view consistency [18, 12]. These works motivate treating support and blend order as part of the rendering model. Our screen-covariance floor is a numerical low-pass regularizer, not an implementation of Mip-Splatting’s complete filtering method.

Allocation and efficient training. 3DGS as MCMC studies population evolution and relocation; Taming 3DGS addresses explicit resource budgets [8, 11]. Structure-aware and error-guided methods, including SAD-GS and TurboGS, allocate computation or primitives using image evidence [10, 3]. Structure-aware densification alone is therefore not a novelty claim here. Our focus is its integration with bounded raster visibility and continuity of learned state in a browser optimizer.

Gaussian splatting on the web. Brush supports cross-platform reconstruction and a browser build; WebDGS implements browser-native differentiable training [2, 17]. WebSplat targets efficient cross-device WebGPU forward rendering, and Visionary combines web splatting with inference [6, 4]. Rendering, training, and preservation of training fidelity are distinct contributions. We evaluate reconstruction quality and recorded optimizer cost, not priority for bringing splatting to the web.

3 Representation and Image Objective

3.1 Coordinate conventions and covariance

Primitive i has world-space mean $\mu_i \in \mathbb{R}^3$, orientation $R_i \in \mathbb{R}^{3 \times 3}$, positive scales s_i , opacity $\alpha_i \in (0, 1)$, and appearance

coefficients. Its covariance is

$$\Sigma_i = R_i \text{diag}(s_{ix}^2, s_{iy}^2, s_{iz}^2) R_i^\top. \quad (1)$$

The implementation description uses logarithmic scales, an opacity logit, and an orthonormalized local frame. Equation 1 describes the represented ellipsoid; it does not require storing a dense covariance.

Let R_c, t_c be world-to-camera extrinsics and $\pi_\kappa : \mathbb{R}^3 \rightarrow \mathbb{R}^2$ the calibrated projection. Then

$$\mathbf{x}_i^c = R_c \mu_i + t_c, \quad \mathbf{u}_i = \pi_\kappa(\mathbf{x}_i^c). \quad (2)$$

The projection Jacobian at the camera-space mean is

$$J_i = \left. \frac{\partial \pi_\kappa}{\partial \mathbf{x}} \right|_{\mathbf{x}=\mathbf{x}_i^c} \in \mathbb{R}^{2 \times 3}. \quad (3)$$

The first-order screen covariance is

$$\Sigma_i^{2D} \approx (J_i R_c) \Sigma_i (J_i R_c)^\top + \beta I_2. \quad (4)$$

Translation changes where J_i is evaluated; it does not multiply the covariance. A homogeneous 4×4 transform cannot be applied directly around a 3×3 covariance.

The described source path adds 0.30 to the screen-covariance diagonals: a variance of 0.30 squared pixels, not a standard deviation of 0.30 pixels. Projection is also conditioned near the camera field-of-view boundary. Non-pinhole paths use finite differences, including different forward and backward stencils. Equation 4 is the ideal local model, not a claim that every implemented camera derivative is exact.

Camera import follows supported COLMAP lens models [13]. Directional color uses SH through degree three: sixteen basis functions per channel, including the constant term. Base color is stored separately from the fifteen nonconstant coefficients. Section 6.3 tests this appearance capacity; camera-model fidelity remains an unablated choice.

3.2 Photometric and structural supervision

For RGB reference I and prediction $\hat{I}$ over pixels Ω , define normalized losses

$$L_1 = \frac{1}{3|\Omega|} \sum_{p \in \Omega} \|\hat{I}(p) - I(p)\|_1, \quad (5)$$

$$L_2 = \frac{1}{3|\Omega|} \sum_{p \in \Omega} \|\hat{I}(p) - I(p)\|_2^2. \quad (6)$$

For a full image at a fixed view, the loss mixture is

$$L_{\text{base}} = (1 - \lambda_s) L_1 + \lambda_s L_{\text{SSIM}}, \quad (7)$$

$$L(t) = (1 - \lambda_2(t)) L_{\text{base}} + \lambda_2(t) L_2. \quad (8)$$

Here $L_{\text{SSIM}} = 1 - \text{SSIM}(I, \hat{I})$ averages an 11×11 Gaussian-window structural measure [16]. The described full-anisotropic path uses $\lambda_s = 0.20$, bounded view confidence, and optional interleaved-tile sampling. Equation 8 isolates the image-loss mixture; effective schedules for the later benchmark switches were not fully logged.

For normalized RGB, per-view PSNR is

$$P_v = -10 \log_{10} \left(\max(\text{MSE}_v, 10^{-12}) \right). \quad (9)$$

The floor reflects the evaluator’s finite-value convention. A scene score averages P_v across evaluated views; the dataset mean then weights the nine scenes equally. This differs in general from computing PSNR on a pooled MSE.

Because MSE defines PSNR, late photometric refinement is a reasonable design choice once structure has formed. It does not follow that a larger λ_2 improves the endpoint of a finite, non-convex optimization. Geometry, opacity, appearance, and density interact; our sweep measures a non-monotonic response.

4 Bounded Differentiable Splatting

4.1 Opacity-relevant support

Let $Q_i = (\Sigma_i^{2D})^{-1}$ and $\mathbf{d}_i = \mathbf{p} - \mathbf{u}_i$. The unclipped alpha contribution is

$$\tilde{a}_i(\mathbf{p}) = \alpha_i \exp \left(-\frac{1}{2} \mathbf{d}_i^\top Q_i \mathbf{d}_i \right). \quad (10)$$

For a threshold τ and $\alpha_i > \tau$, the ideal relevant footprint is

$$\Omega_i(\tau) = \{ \mathbf{p} : \mathbf{d}_i^\top Q_i \mathbf{d}_i \leq 2 \log(\alpha_i/\tau) \}. \quad (11)$$

The described path rejects subthreshold contributions, uses $\tau = 1/255$, and caps accepted alpha at 0.999. At $\alpha_i = \tau$, support can degenerate to a point. Frustum tests, small-radius rejection, and pixel sampling further constrain practical support; the renderer is not an unlimited continuous Gaussian field.

Training divides the image into 8×8 tiles. A broad-phase bounding box limits candidate tiles. Analytic intersection of the ellipse with each horizontal tile slab then determines a horizontal interval. For a tile rectangle B , the ideal admission test is equivalently

$$\min_{\mathbf{p} \in B} (\mathbf{p} - \mathbf{u}_i)^\top Q_i (\mathbf{p} - \mathbf{u}_i) \leq 2 \log(\alpha_i/\tau). \quad (12)$$

Figure 1 shades tiles using this criterion. Unlike a center-only test, it admits tiles crossed by a long oblique splat. A conservative bounding box also retains support, but admits extra work; geometric admission reduces that work before capacity limiting.

4.2 Finite retention and stable order

Each tile distributes candidates across logarithmic depth slices and hash partitions. Fixed primary and overflow regions bound temporary storage. Overload invokes seeded replacement rather than always keeping the earliest arrivals. A fixed seed does not make concurrent replacement deterministic or unbiased: membership can depend on execution order. We do not treat the retained composite as a Monte Carlo estimator of the unbounded one.

Retained entries are sorted by camera depth, with Gaussian identity breaking ties. A bounded merge supplies the sequence traversed in both passes. This gives a consistent order for a given retained set, but cannot recover discarded contributors.

Retained-intersection coverage and overflow counts therefore accompany quality metrics.

At a pixel, index accepted, retained contributors from near to far. With the thresholded and capped alpha a_i ,

$$T_1 = 1, \quad T_{i+1} = T_i(1 - a_i), \quad (13)$$

$$\hat{\mathbf{C}} = \sum_{i=1}^n T_i a_i \mathbf{c}_i + T_{n+1} \mathbf{C}_{\text{bg}}. \quad (14)$$

Alpha compositing is order dependent. Reusing the sequence prevents the passes from using different retained orders; it does not establish exact per-pixel visibility or equivalence to another rasterizer.

4.3 Reverse compositing

The reverse pass maintains a compact suffix state instead of storing every intermediate color at every pixel:

$$\mathbf{B}_{n+1} = \mathbf{C}_{\text{bg}}, \quad \mathbf{B}_i = a_i \mathbf{c}_i + (1 - a_i) \mathbf{B}_{i+1}. \quad (15)$$

For fixed membership and order, away from threshold and clipping boundaries,

$$\frac{\partial \hat{\mathbf{C}}}{\partial \mathbf{c}_i} = T_i a_i \mathbf{I}_3, \quad (16)$$

$$\frac{\partial \hat{\mathbf{C}}}{\partial a_i} = T_i (\mathbf{c}_i - \mathbf{B}_{i+1}). \quad (17)$$

The image-loss gradient contracts these Jacobians into parameter derivatives. The equations do not differentiate discrete membership. Exact differentiation of a saturated alpha cap gives zero derivative with respect to its unclipped input; threshold crossings are nondifferentiable. The shader’s branch handling and camera stencils require kernel-level checks.

Per-tile contributions are reduced before global publication. Optional subgroup reductions have a workgroup-memory fallback. A Gaussian-wise projection-backward pass maps the screen-space payload into position, scale, orientation, opacity, and SH updates. This removes expensive per-Gaussian algebra from the pixel-overlap loop, without implying full-gradient equality to a CUDA reference.

4.4 Integer gradient transport

Core WGSL provides integer atomic accumulation [15]. The described optimizer-gradient transport uses scale $S = 2^{27}$, with a publication helper represented by

$$q_k = \text{round}(S \text{clip}(g_k, -8, 8)), \quad \hat{\mathbf{g}} = S^{-1} \sum_k q_k. \quad (18)$$

The decoded sum has this ordinary arithmetic interpretation only while the signed accumulator stays in range. Pixel-row contributions are additionally clipped to $[-0.24, 0.24]$. Selected screen-space gradients are conditioned by 2^{10} before transport and unscaled afterward. Evidence integrals use a separate scale and are not optimizer gradients.

Absent clipping, nonfinite inputs, and accumulator overflow, nearest rounding gives

$$\left| S^{-1} \sum_{k=1}^K \text{round}(S g_k) - \sum_{k=1}^K g_k \right| \leq \frac{K}{2S}. \quad (19)$$

This is a conditional quantization bound, not a quality guarantee. At $S = 2^{27}$, an i32 accumulator spans $[-16, 16-2^{-27}]$. Clipping each contribution to eight does not keep the sum in range: two positive contributions of eight already exceed the maximum. The archive does not establish universal overflow detection or safe accumulation throughout training.

5 Population and Browser Execution

5.1 Preserving learned state

Density control combines residual evidence, structural response, projected extent, visibility, and contribution history. A residual histogram supplies a view-relative threshold rather than one fixed radiometric cutoff. Allocation also accounts for whether new primitives can remain visible to the bounded rasterizer: growing the population under inadequate retained coverage can add parameters that are inconsistently trained.

The described population update carries forward SH appearance and relevant Adam state [9]. For parent opacity α_p and m co-located children, initialization uses

$$\alpha_c = 1 - (1 - \alpha_p)^{1/m}. \quad (20)$$

Thus $(1 - \alpha_c)^m = 1 - \alpha_p$, preserving coincident peak transmittance. This identity does not preserve the full spatial footprint after Gaussian attenuation, scaling, displacement, or alpha clipping. It is a continuity heuristic, not radiance or integrated-mass conservation.

The evaluated configuration avoids recurrent global opacity resets. State inheritance and local opacity continuity aim to reduce disruption while adding capacity. Their independent PSNR contributions have not been measured.

5.2 A local optimization loop

Images and calibrated cameras enter a bounded resident-image cache. Adapter limits determine compatible allocations. Training is submitted in packets that allow interaction between submissions, and queue synchronization defines timing boundaries. The Gaussian optimizer requires no server-side fallback, although input camera poses may have been produced externally.

The described checkpoint format persists the Gaussian field and schedules, but not complete Adam moments. Resumed training therefore need not reproduce an uninterrupted trajectory. Finished fields can be exported as Gaussian PLY, SOG, or OpenUSD for downstream use.

6 Evaluation

6.1 Evidence and protocol

The development archive comprises seven exports and 350 run rows. Deduplicating by run ID leaves 321 unique records: 300 completed, 18 failed, and three cancelled. These status counts describe a mixed development archive, not a controlled reliability trial. All 29 repeated rows have matching metric payloads; export-specific signed media URLs are not part of that equality. The archive spans multiple builds and workloads, rather than repeated trials of a single configuration.

The primary cohort consists of three complete nine-scene suites sharing recorded commit 3fcad81ff8e8, one deploy-

ment ID, and one benchmark-worker SHA-256. Selection follows the suite labels listed in the companion evidence, not per-scene best scores. Four additional runs at the same recorded commit are excluded by this rule.

Build provenance. The 27 runs contain eight aggregate asset fingerprints and differing observed asset inventories. Shared listed assets have matching byte counts and hashes; the union contains fourteen asset paths. The record therefore supports a common recorded deployment and worker, but not a complete byte-for-byte execution manifest. Effective tail coefficients and a commit-matched source snapshot for the later switches are not fully available.

All primary runs use 30,000 iterations, a one-million-Gaussian cap and seed cap, resolution divisor two, no adaptive resolution, and every-eighth-image holdout on Mip-NeRF 360 [1]. A seed cap limits initial population size; it is not evidence of matched random seeds. PSNR and SSIM follow the same named held-out protocol, whose described metric path operates on normalized sRGB-domain values. LPIPS was not recorded. The companion tables retain run IDs, settings, view counts, and logged render-canvas dimensions. Those canvas dimensions do not establish the dimensions of every evaluated image.

We denote the reference by B_0 , the combined-tail request by B_T , and the optimizer-tail request by B_O . All run the WebGPU trainer. The word “native” in internal switch names does not designate a CUDA baseline. These comparisons concern requested configurations, not independently verified changes at every optimizer step.

All primary runs use the RTX 4090 GPU class, but CPU hosts differ across suites. A common physical device and matched random seeds are not established. With one run per scene and condition, we report descriptive differences, not confidence intervals, statistical significance, or cross-configuration timing speedups. Dataset identity alone does not make the results comparable to published 3DGS scores obtained at different resolutions or with different metric settings [5].

6.2 Nine-scene reconstruction quality

Table 1 reports 27.701 dB and 0.7987 SSIM for B_0 , 27.741 dB and 0.8013 for B_T , and 27.762 dB and 0.7993 for B_O . The reference already delivers most of the measured fidelity. The tail requests provide incremental, scene-dependent changes rather than accounting for the overall reconstruction quality.

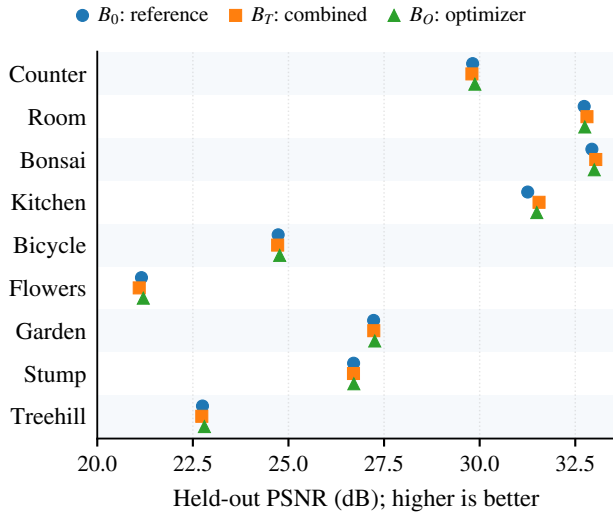
The optimizer-tail request increases observed PSNR on all nine scenes, averaging +0.061 dB. Kitchen has the largest increase, 0.237 dB. SSIM increases on seven scenes and decreases slightly on Bonsai and Treehill. The combined-tail request increases SSIM on all nine scenes but PSNR on only four: Kitchen gains 0.296 dB, while Counter and Flowers lose 0.019 and 0.058 dB, respectively. Figure 2 separates absolute quality from these scene-wise differences.

Bonsai reaches 33.036 dB under B_T . The complete dataset mean also includes Flowers and Treehill near 21.1 and 22.8 dB, retaining the more demanding results rather than substituting a selective development average. Final populations average approximately 0.886 million Gaussians across the 27 runs.

Table 1: Complete nine-scene held-out results. Each column is a WebGPU request configuration at 30k iterations and a 1M cap. The suites share recorded commit, deployment, and worker identifiers; asset-manifest qualifications are given in Section 6.1. Means weight scenes equally. These are not external baselines.

Scene	B_0 : reference		B_T : combined tail		B_O : optimizer tail	
	PSNR (dB)	SSIM	PSNR (dB)	SSIM	PSNR (dB)	SSIM
Counter	29.816	0.9247	29.797	0.9267	29.874	0.9248
Room	32.733	0.9396	32.806	0.9407	32.749	0.9397
Bonsai	32.934	0.9495	33.036	0.9507	32.997	0.9495
Kitchen	31.256	0.9336	31.552	0.9385	31.494	0.9365
Bicycle	24.731	0.7016	24.717	0.7054	24.770	0.7025
Flowers	21.156	0.5490	21.097	0.5522	21.204	0.5496
Garden	27.227	0.8148	27.231	0.8180	27.256	0.8155
Stump	26.703	0.7612	26.700	0.7633	26.709	0.7616
Treehill	22.751	0.6147	22.733	0.6166	22.802	0.6142
Mean	27.701	0.7987	27.741	0.8013	27.762	0.7993

(a) Absolute held-out PSNR



(b) Change from reference

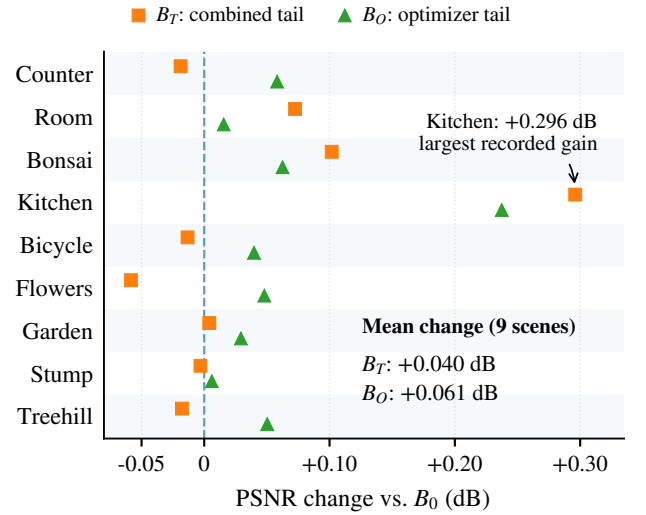


Figure 2: Absolute quality and differences from the reference. (a) Held-out PSNR for all three request configurations. (b) Scene-wise PSNR differences from B_0 . Small vertical offsets separate configurations within each scene. Differences use unrounded run values. Markers show recorded endpoints, not uncertainty estimates; one run per condition does not establish significance.

6.3 What preserves PSNR?

Directional appearance capacity. A separate Counter pair uses one recorded build, a common 30k iteration budget, 1M cap, divisor-two resolution, and RTX 5080 hardware class, with maximum SH degree three or one. Final telemetry confirms the active degrees. PSNR falls from 29.814 to 29.058 dB, and SSIM from approximately 0.9249 to 0.9150. The observed 0.756 dB difference supports retaining directional appearance capacity on this scene. One run per condition does not establish a universal effect size or eliminate all uncontrolled run variation.

Late photometric refinement. Another fixed-recorded-build Counter sweep compares two default repeats with requested late-L2 targets of 0.925, 0.95, and 1.0. The default mean is 29.818 dB and 0.9247 SSIM. Target 0.95 reaches 29.874 dB with lower SSIM (0.9237); target 1.0 drops to 29.695 dB and 0.9154 SSIM. Figure 3 shows discrete endpoints. The SH

pair and L2 sweep use separate builds, both distinct from the primary suites.

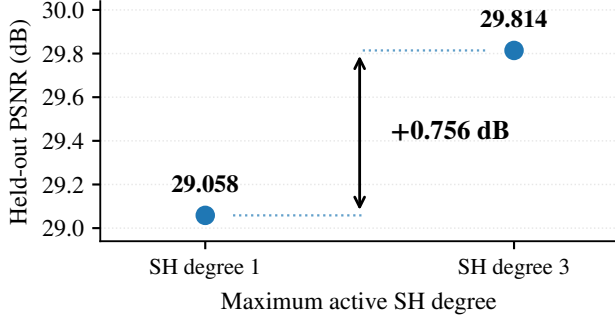
Two practical lessons emerge from these experiments. First, retaining degree-three appearance capacity is supported by the measured Counter pair. Second, more aggressive MSE weighting is not automatically better: objective settings must be judged against both PSNR and SSIM. These are findings from the recorded experiments, not universal tuning rules. Camera projection, tile admission, quantization, and state-preserving densification still require separate component studies; requested targets do not establish an identical effective coefficient at every step.

6.4 Optimizer time and bounded visibility

All nine reference runs disable formation recording, continuous capture, and walkthrough generation. Each is completed, reports synchronized, clean, valid timing, and has empty timing-qualification, protocol-violation, and GPU-concern lists. Under

(a) SH degree comparison

Counter; one recorded run per condition

**(b) Late-L2 target sweep**

Counter; separate endpoint measurements

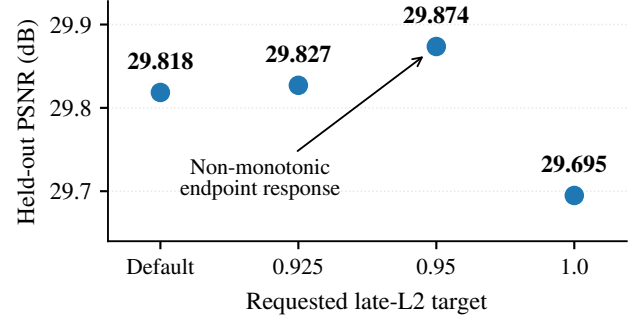


Figure 3: Counter quality interventions. (a) The recorded SH degree-three versus degree-one pair differs by 0.756 dB. (b) Requested late-L2 targets have a non-monotonic endpoint response. Default averages two runs; each other condition has one. The panels use distinct builds and show neither an interpolated training trajectory nor camera-matched visual comparisons.

this explicit gate, their optimizer times range from 8.06 to 17.38 minutes, with a scene mean of 12.60 minutes (Figure 4). These are recorded optimizer durations, not upload-to-delivery latency, rendering frame rate, or speedups over another trainer.

Reference runs report 8,192 retained slots per tile. Final retained-intersection coverage ranges from 97.160% on Counter to approximately 100% on the least congested scenes. Counter records 68,634 overflowed buckets. High-quality completion is therefore not evidence of lossless retention. Coverage is a logged snapshot rather than the minimum across all training views; rounding to 100.00% is not a universal zero-overflow guarantee.

Full-device peak telemetry spans approximately 4.53–7.58 GiB when the logged memory values are divided by 1,024. It is not process-isolated memory, and estimated buffer sizes omit resources. Cross-configuration speed comparisons would be confounded by differing hosts and devices; one optimizer-tail Stump run also records a throttling concern.

Across the unique-run archive, 80 completed records pass the telemetry-only gate. Requiring all three media-generation settings to be disabled leaves 78. These records still span different workloads and builds and are not pooled into a speed claim.

6.5 Visual evidence and provenance

Figure 5 shows the archived final previews from all nine reference runs, in the same scene order as Table 1. The supplied media bundle provides the actual JPEGs and a manifest linking their short filenames to full run IDs and stable benchmark object keys. All nine imported files match the byte counts recorded in the benchmark exports. The companion manifest records their paths, dimensions, and SHA-256 hashes; these locally computed hashes identify the imported files rather than independently authenticating the remote originals.

Each image is embedded at its original 1920×1080 resolution and scaled for page layout without cropping, recoloring, or recompression. The panel metrics are held-out scene averages, not scores of the pictured view. Preview-camera identities and ground-truth pairings are not established, so the figure

Reference-cohort optimizer time

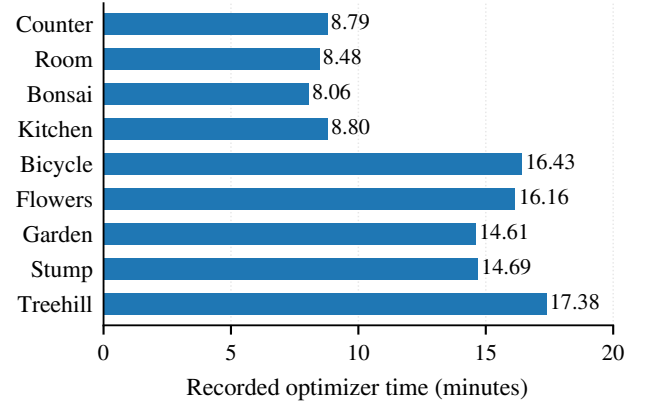
B₀; media disabled; scene mean 12.60 min

Figure 4: Reference-cohort optimizer time. All nine RTX 4090-class runs pass the explicit timing and media-disabled gate. Their mean is 12.60 minutes; differing image dimensions and scene complexity remain part of the workload.

supplies qualitative run-level evidence rather than pixelwise comparisons or visual ablations.

7 Discussion and Limitations

7.1 What the measurements establish

The retained results establish a working browser-native Gaussian optimizer with measured image quality, optimizer duration, and visibility-capacity telemetry. All nine benchmark scenes are represented in each primary suite. The original previews make the resulting fields inspectable, while the separate SH and loss experiments connect specific development decisions to observed quality changes.

The implementation combines several resource-conscious approximations. Capacity limits can omit contributors, first-order covariance projection and numerical camera stencils approximate image formation, integer transport has precision and range limits, and peak-opacity matching preserves only local continuity. These mechanisms are explicit parts of the



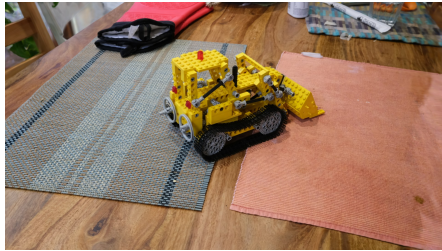
(a) Counter
29.816 dB SSIM 0.9247
Optimizer 8.79 min



(b) Room
32.733 dB SSIM 0.9396
Optimizer 8.48 min



(c) Bonsai
32.934 dB SSIM 0.9495
Optimizer 8.06 min



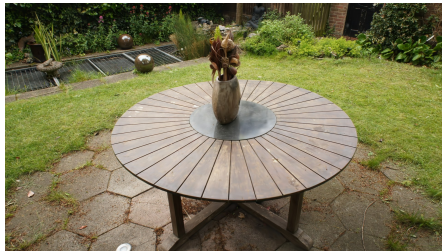
(d) Kitchen
31.256 dB SSIM 0.9336
Optimizer 8.80 min



(e) Bicycle
24.731 dB SSIM 0.7016
Optimizer 16.43 min



(f) Flowers
21.156 dB SSIM 0.5490
Optimizer 16.16 min



(g) Garden
27.227 dB SSIM 0.8148
Optimizer 14.61 min



(h) Stump
26.703 dB SSIM 0.7612
Optimizer 14.69 min



(i) Treehill
22.751 dB SSIM 0.6147
Optimizer 17.38 min

Figure 5: Mip-NeRF 360 reference-run reconstructions. Archived final previews from all nine B_0 runs in Table 1, embedded as full 1920×1080 images with their native aspect ratio preserved. PSNR and SSIM are held-out scene means, not measurements of the shown frame; time is the recorded optimizer duration. These run-level previews are not camera-matched ground-truth comparisons. Full run IDs and file hashes are retained in the companion manifest.

design, not evidence of mathematical identity with an unlimited native rasterizer.

The evidence package supports arithmetic, run-level telemetry, and media-provenance checks. Differing observed asset inventories, incomplete effective-parameter logging, and a method description that predates the later tail switches limit source-level attribution. The present record is not an independently replicated browser-versus-native comparison or a verification of every GPU derivative.

7.2 A controlled extension

A controlled comparison requires fixed browser and native implementations, identical cameras and decoded images, a shared split, resolution, and metric code, matched hardware, and repeated random seeds. Synchronized timing with media capture disabled should be retained. LPIPS and camera-matched reference crops would extend the current PSNR/SSIM

evaluation [19]; unmatched previews cannot supply those comparisons.

Separate studies should evaluate camera-model choices, tile admission, retention capacity, SH capacity, objective schedules, and state inheritance. Overflow and gradient-range tests, build-matched source, effective-parameter logs, and kernel-level finite-difference tests would strengthen reproducibility. Existing CPU checks cover selected mathematical identities and local derivatives; they do not execute WGSL or certify the complete gradient path. An untouched validation set is also needed after repeated development-scene tuning.

The results cover the tested desktop GPU configurations. Cross-browser parity, dynamic scenes, color-managed HDR correctness, and learned camera refinement remain outside the demonstrated scope. Broader device qualification and end-to-end workflow measurement are separate from the recorded optimizer durations; exact checkpoint continuation also remains unverified.

8 Conclusion

Splatfiction establishes the Gaussian-training stage of a browser-native 3D creation workflow. It retains anisotropic shape, degree-three directional appearance, an explicit image objective, and controlled population refinement while making visibility capacity a measurable resource constraint.

Across the complete nine-scene suites, retained results reach 27.70–27.76 dB mean held-out PSNR at 30,000 iterations and a one-million-Gaussian cap. The reference records 8.06–17.38 minutes of optimizer time per scene. Our Counter experiments support retaining appearance capacity and evaluating late-stage objective settings as a quality tradeoff rather than a one-directional improvement.

The product foundation is browser-side training with reusable Gaussian PLY, SOG, and OpenUSD outputs. The next stage is to broaden and harden that capability through repeatable comparisons, numerical qualification, and deployment coverage. This separates the working technology demonstrated today from the additional validation needed for wider deployment.

References

- [1] Jonathan T. Barron, Ben Mildenhall, Dor Verbin, Pratul P. Srinivasan, and Peter Hedman. Mip-NeRF 360: Unbounded anti-aliased neural radiance fields. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5470–5479, 2022.
- [2] Arthur Brussee and contributors. Brush: 3D reconstruction for all. <https://github.com/ArthurBrussee/brush>, 2026. Software repository; accessed 22 September 2026.
- [3] Zheng Dong, Daifei Qiu, Pinxuan Dai, Ke Xu, Jiamin Xu, Lili He, Rynson W. H. Lau, and Weiwei Xu. TurboGS: Accelerating 3D gaussian splatting via error-guided sparse pixel sampling and optimization. arXiv:2606.15924, 2026.
- [4] Yuning Gong, Yifei Liu, Yifan Zhan, Muyao Niu, Xueying Li, Yuanjun Liao, Jiaming Chen, Yuanyuan Gao, Jiaqi Chen, Minming Chen, et al. Visionary: The world model carrier built on WebGPU-powered gaussian splatting platform. arXiv:2512.08478, 2025.
- [5] GRAPHDECO Research Group. 3D gaussian splatting: Reference implementation. <https://github.com/graphdeco-inria/gaussian-splatting>, 2026. Software documentation; accessed 22 September 2026.
- [6] Yudong Han, Chao Xu, Xiaodan Ye, Weichen Bi, Xuanhuai Xu, Zilong Dong, and Yun Ma. WebSplat: Efficient and faithful in-browser 3D gaussian splatting across devices via WebGPU. In *Proceedings of the 34th ACM International Conference on Multimedia*, 2026. doi: 10.1145/3767308.3835220.
- [7] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 3D gaussian splatting for real-time radiance field rendering. *ACM Transactions on Graphics*, 42(4), 2023. doi: 10.1145/3592433.
- [8] Shakiba Kheradmand, Daniel Rebain, Gopal Sharma, Weiwei Sun, Yang-Che Tseng, Hossam Isack, Abhishek Kar, Andrea Tagliasacchi, and Kwang Moo Yi. 3D gaussian splatting as Markov Chain Monte Carlo. In *Advances in Neural Information Processing Systems*, 2024.
- [9] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations*, 2015.
- [10] Linjie Lyu, Ayush Tewari, Jianchun Chen, Thomas Leimkühler, and Christian Theobalt. Faster 3D gaussian splatting convergence via structure-aware densification. In *ACM SIGGRAPH 2026 Conference Papers*, 2026. doi: 10.1145/3799902.3811212.
- [11] Saswat Subhrajyoti Mallick, Rahul Goel, Bernhard Kerbl, Francisco Vicente Carrasco, Markus Steinberger, and Fernando De La Torre. Taming 3DGS: High-quality radiance fields with limited resources. In *SIGGRAPH Asia 2024 Conference Papers*, 2024. doi: 10.1145/3680528.3687694.
- [12] Lukas Radl, Michael Steiner, Mathias Parger, Alexander Weinrauch, Bernhard Kerbl, and Markus Steinberger. StopThePop: Sorted gaussian splatting for view-consistent real-time rendering. *ACM Transactions on Graphics*, 43(4), 2024.
- [13] Johannes L. Schönberger and Jan-Michael Frahm. Structure-from-motion revisited. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4104–4113, 2016.
- [14] W3C GPU for the Web Working Group. WebGPU. Technical report, World Wide Web Consortium, 2026. Technical specification. <https://www.w3.org/TR/webgpu/>.
- [15] W3C GPU for the Web Working Group. WebGPU shading language. Technical report, World Wide Web Consortium, 2026. Technical specification. <https://www.w3.org/TR/WGSL/>.
- [16] Zhou Wang, Alan C. Bovik, Hamid R. Sheikh, and Eero P. Simoncelli. Image quality assessment: From error visibility to structural similarity. *IEEE Transactions on Image Processing*, 13(4):600–612, 2004. doi: 10.1109/TIP.2003.819861.
- [17] WebDGS contributors. WebDGS: Differentiable gaussian splatting training and viewing in WebGPU. <https://github.com/krispy-kenay/WebDGS>, 2026. Software repository; accessed 22 September 2026.
- [18] Zehao Yu, Anpei Chen, Binbin Huang, Torsten Sattler, and Andreas Geiger. Mip-Splatting: Alias-free 3D gaussian splatting. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 19447–19456, 2024.
- [19] Richard Zhang, Phillip Isola, Alexei A. Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 586–595, 2018.
- [20] Matthias Zwicker, Hanspeter Pfister, Jeroen van Baar, and Markus Gross. EWA volume splatting. In *Proceedings of IEEE Visualization*, pages 29–36, 2001.