Counting Contributions, Not Views: Occlusion-Aware Densification for 3D Gaussian Splatting

Marcel Rogge[1,2], René Schuster[1,2], and Didier Stricker[1,2]

[1] University Kaiserslautern-Landau – RPTU
[2] German Research Center for Artificial Intelligence – DFKI

Abstract. 3D Gaussian Splatting grows a dense scene from a sparse initialization through adaptive density control (ADC), driven by per-Gaussian statistics gathered during rendering. These statistics pass a frustum-based visibility filter blind to occlusion: fully occluded Gaussians still count as observed, diluting the averaged gradient signal and suppressing densification precisely where geometry is occluded. We correct this with an occlusion-aware filter that restricts densification statistics to views in which a Gaussian actually contributes, and replace the criterion's raw view-count normalizer with an average-contribution one. Integrated into a StopThePop-based framework, the two together improve every image-quality metric over the occlusion-blind baseline on occlusion-heavy scenes (up to 6% lower LPIPS), at less than half the Gaussians a view-count normalizer requires. Under the longer schedules these large scenes require, the count stays bounded where most criteria diverge, and on smaller benchmarks our method stays competitive at a fraction of the competitors' count.

Keywords: Gaussian Splatting · Adaptive Density Control · Densification · 3D Reconstruction · Large-Scale Scenes · Novel View Synthesis

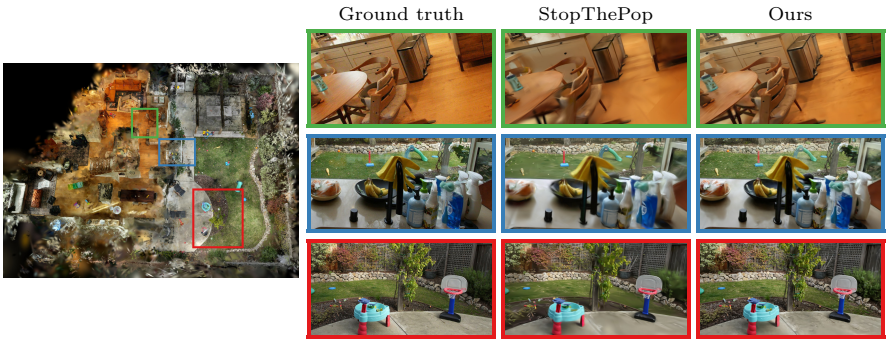


Fig. 1: Occlusion-aware densification. Left: top-down overview of the large alameda scene (modified reconstruction; for orientation only). Most geometry is occluded from any given position. Right: crops of the marked locations. Occlusion-blind densification (such as StopThePop) systematically *starves* occluded regions of Gaussians; our occlusion-aware approach resolves the missing detail.

1 Introduction

3D Gaussian Splatting (3DGS) [13] represents a milestone in 3D scene reconstruction and novel-view synthesis. It introduced a large multi-component system, with a tile-based rasterizer and adaptive density control (ADC) purpose-built for the new representation. Follow-up works improve these components largely in isolation, many focusing on ADC, which is essential for growing a dense reconstruction from the sparse initialization. However, we find that the accumulation of the densification statistics at the root of all ADC decisions has a fundamental flaw: the frustum-based visibility filter that gathers them tests only whether a Gaussian could appear in a view, not whether it actually contributes to the rendered image. A fully occluded Gaussian yields zero gradients, yet the filter still counts its view in the criterion's denominator, dragging the average down and stalling densification wherever geometry is occluded. The resulting under-densification grows with the degree of occlusion and degrades quality most where high detail and occlusion co-occur. Large, occlusion-heavy scenes such as multi-room apartments are affected most severely (Fig. 1). We correct this with an occlusion-aware visibility filter that updates a Gaussian's statistics only from views in which it is actually rendered; densification then reaches occluded regions whenever their level of detail requires it.
We further find that normalizing the accumulated gradient by a raw view count does not match the contribution-weighted nature of the tracked gradients. We derive a more natural normalizer that accumulates a Gaussian's average contribution to each rendered view. To validate our occlusion-aware densification, we build a framework around StopThePop [20] that consolidates established densification improvements from separate prior methods into a strong baseline. We evaluate both changes on this baseline and directly on StopThePop. On the occlusion-heavy Zip-NeRF [2] scenes, the resulting method outperforms StopThePop on all image-quality metrics, reducing LPIPS by up to 12%. Its Gaussian count also stays bounded under the longer schedules these large scenes require, where most competing criteria diverge. On smaller benchmarks, it performs comparably to the state of the art with up to 63% fewer Gaussians. Our contributions are as follows:

- To the best of our knowledge, we are the first to replace the visibility filter itself: the frustum-based test that underlies *all* densification statistics. Our occlusion-aware filter updates a Gaussian's statistics only from views in which it is actually rendered, independent of the criterion built on top.
- We introduce a contribution-based densification criterion that is more closely related to the gradient signal used by ADC, leading to better densification decisions, demonstrated on small- and large-scale scenes.
- We combine both contributions with established densification improvements and other refinements into a strong framework that does not require per-scene tuning and has fewer hyperparameters (*e.g.* eliminating the unstable scene extent). We will make the code available to the research community.

2 Background and Related Work

2.1 Adaptive Density Control

3D Gaussian Splatting [13] represents a scene as a set of 3D Gaussians, each defined by a 3D mean position, a 3D covariance matrix (parameterized by a rotation and per-axis scale), an opacity o, and a view-dependent color c derived from spherical harmonics [7]. For rendering, each Gaussian is projected onto the image as a 2D footprint with a screen-space mean and radius. Starting from a sparse point cloud, 3DGS grows and prunes its set of Gaussians via adaptive density control (ADC), executed every 100 iterations after a 500-iteration warm-up and disabled after the first half of training. A scene extent E, computed from the training cameras, provides the reference scale for several ADC decisions.

Criterion. During rendering, a frustum-based visibility filter accumulates per-Gaussian statistics: the 2D screen-space positional gradient $g_{i,v}$ of Gaussian i in view v, the screen-space radius, and a view count M. At each ADC event, a Gaussian is marked for densification if its accumulated gradient magnitude, averaged over the view count, exceeds a predefined threshold (0.0002 by default).

Densification. To resolve the problem of an insufficient number of Gaussians capturing a region's textural or structural detail (from here on described as under-densification), a marked Gaussian is densified in one of two modes: if it is large relative to the scene extent, it is *split* into two downscaled children sampled from the parent's distribution, otherwise it is *cloned*, *i.e.* duplicated, in place.

Pruning. Redundant or noisy Gaussians are removed using pruning mechanisms. An opacity prune removes every Gaussian whose opacity o_i falls below a predefined threshold (0.005). A size-based prune removes Gaussians whose screen-space radius or world-space scale exceeds a fixed threshold. To clear floater artifacts, 3DGS additionally resets all opacities to near zero (0.01) every 3000 iterations; important Gaussians recover quickly, floaters stay dim and fall to the opacity prune. StopThePop [20] offers a gradual opacity *decay* as an alternative to the drastic reset. We evaluate both as the (Reset) and (Decay) variants.

2.2 Adaptive Density Control Improvements

All of the following ADC improvements inherit the frustum-based visibility filter described above, which counts every in-frustum Gaussian, occluded or not. We make this filter occlusion-aware (Sec. 3.1).

Densification Criterion. AbsGS [28] and GOF [29] counter gradient cancellation in the accumulation, which otherwise causes under-densification of high-detail regions. PixelGS [31] instead questions the normalization by the observation count and proposes a pixel-count-weighted mean. Grubert *et al.* [9] adopt the PixelGS strategy with an ascending threshold that shifts densification earlier.

Taming3DGS [18] replaces the threshold with a per-Gaussian multi-term score under a fixed densification budget. However, budget-controlled methods require a target count as input, a value that is not known and that a good densification criterion should find. We propose to normalize a Gaussian's accumulated gradient by the accumulated average contributions of its rendered views (Sec. 3.2), rather than the observation count of 3DGS or the visible-pixel count of PixelGS. We also adopt GOF's absolute-gradient signal against cancellation.

Densification Step (Split & Clone). GOF observes that clones remain clustered and instead samples the child's position from the parent's distribution, while Bulò *et al.* [22] show that the clone operation introduces an opacity bias. EFA-GS [23] and Deng *et al.* [4] drop the clone entirely, and Deng *et al.* additionally propose a geometrically more meaningful split procedure. Grubert *et al.* address inconsistencies in the scene-extent computation that governs some ADC decisions by also accounting for the input point cloud. Other works replace the densification step more radically [3,6,15,24]. Our experiments also support dropping the clone: we notice large floater clusters due to excessive cloning on scenes where the scene extent is not well matched to the size of detail (Tab. 4). We therefore follow the split-only approach of EFA-GS and Deng *et al.*, including the latter's split procedure. This removes one dependence on the scene extent.

Pruning. Grubert *et al.* restrict the opacity prune to Gaussians that also contribute little; EAGLES [8] and OC2DGS [21] prune low- or non-contributing Gaussians at intervals long enough to observe every training view; ReLaGS [26] and RadSplat [19] prune by maximum contribution over a full pass, at isolated points of the optimization. We prune by maximum contribution, as in ReLaGS and RadSplat, but throughout the optimization, as in EAGLES and OC2DGS.

2.3 Large-Scale Scenes

Large-scale scenes harbor a range of problems for Gaussian Splatting: Kerbl *et al.* [14] partition them into chunks optimized individually after a coarse full-scene pass. They note problems with the regular densification regime and adopt the maximum 2D screen-space gradient as the criterion. OccluGaussian [17] makes the partitioning occlusion-aware, clustering by what each camera sees so that no chunk contains heavy occlusions, and optimizes chunks individually before merging them. RadSplat observes that lighting variations in the Zip-NeRF dataset [2] cause 3DGS to struggle and produce overly smooth results even with an exposure-handling module. SMERF [5] performs extensive densification-related hyperparameter tuning on Zip-NeRF yet reports low Gaussian counts, indicating under-densification. Lowering the gradient threshold raises the count only at the cost of more floaters. Neither work identifies the root cause. We argue it is occlusion-blind densification (a cause that exposure correction and hyperparameter tuning leave unaddressed) and provide supporting evidence in Sec. 4.3. Unlike OccluGaussian, we do not avoid this problem by optimizing occluded regions as individual small scenes, but instead make the densification occlusion-aware, enabling a single full-scene optimization.

3 Method

We build on StopThePop [20], a state-of-the-art Gaussian Splatting renderer whose exact per-pixel depth ordering removes popping artifacts and keeps occlusion boundaries intact. Hybrid transparency [10], in contrast, sorts only the front-most Gaussians and composites all remaining ones without enforcing occlusion ordering. Leaving even fully occluded Gaussians with a nonzero contribution stands in opposition to our motivation of occlusion-aware densification. However, our method is not dependent on the per-pixel sorting of StopThePop: it relies only on the per-pixel blending weights that virtually every Gaussian Splatting rasterizer already computes. Our contributions can therefore transfer to other bases (evaluated on 2DGS [12] and 3DGS [13] in Sec. A.2 of the supplementary material).

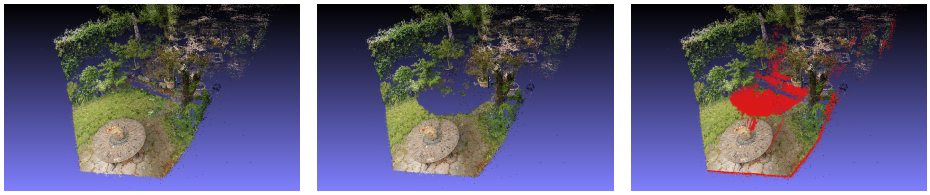
Differentiable rasterization. Projected onto the image plane, a Gaussian i forms a 2D footprint $G_{i,p} \in (0, 1]$ over pixels p [32], yielding the per-pixel opacity $\alpha_{i,p} = o_i G_{i,p}$. 3DGS renders with a tile-based differentiable rasterizer: a preprocessing step culls Gaussians against the camera view frustum, maps the remaining ones to the image tiles they overlap, and sorts them by view-space depth. Each pixel p is rendered by front-to-back alpha compositing of the depth-sorted Gaussians that overlap it, yielding its color C_p:

$$C_p = \sum_i c_i \, \alpha_{i,p} \, T_{i,p} + c_{bg} \, T_{final,p}, \qquad T_{i,p} = \prod_{j<i} (1 - \alpha_{j,p}), \tag{1}$$

where the transmittance $T_{i,p}$ is the fraction of light that reaches Gaussian i at pixel p past the Gaussians in front of it. Rendering terminates early once $T_{i,p}$ saturates (falls below 10^{-4}), so Gaussians behind that point are never rendered. The remaining transmittance $T_{final,p}$ is filled with a constant background color c_{bg}. The product $w_{i,p} = \alpha_{i,p} \, T_{i,p}$ is the blending weight with which Gaussian i contributes to pixel p: its per-pixel opacity $\alpha_{i,p}$ weighted by its visibility $T_{i,p}$. The complete integrated framework, which assembles established densification improvements around them, is described in Sec. 4.2.

3.1 Occlusion-Awareness

As discussed in Sec. 2.1, densification-related statistics are gathered via a visibility filter based on a view-frustum test. This reflects potential, not actual, observations of a Gaussian. From here on, we refer to Gaussians that are not rendered at all as *occluded*: transmittance saturates before reaching them at every pixel they cover. With a frustum-based filter, occluded Gaussians thus enter the accumulated densification statistics despite not participating in the rendered image. Although their gradients are zero, every occluded view still increases the criterion's observation count and updates the maximum-radius tracking. Increasing the observation count results in a downward pressure on the densification criterion of any Gaussian that lies within the frustum while being obstructed.



(a) Frustum-visible (b) Contributing (c) Occluded (red)

Fig. 2: Occlusion-aware versus frustum-based visibility filter. Showing all Gaussians in front of a camera from the Mip-NeRF360 garden scene, which (a) pass the frustum test or (b) actually contribute to the rendered image. (c) shows their difference in red.

The consequence is under-densification in occlusion-heavy scenes such as the multi-room apartments of Zip-NeRF [2], as also observed by prior work [5, 19]. We propose an occlusion-aware visibility filter that restricts the influence of rendering statistics to those Gaussians that actually participate in rendering the image. For the filter itself, a binary signal suffices: a Gaussian counts as observed in a view only if it contributes to at least one pixel. We obtain this signal from the total contribution the Gaussian accumulates during rendering: the sum of its blending weights $w_{i,p}$ over all pixels of the view. This quantity is reused by our criterion in Sec. 3.2. If the total contribution is zero, the Gaussian did not participate in the rendering and is therefore occluded (Fig. 2). Replacing the original frustum-based visibility filter with our occlusion-aware filter thus ensures occlusion-aware densification statistics.
The occlusion signal could equally come from a Boolean rendered flag as computed in OC2DGS [21] or a rendered-pixel count as in PixelGS [31]. However, it is essential that the filter gating the densification statistics is derived from such a rendering signal rather than remaining based on the view frustum.

3.2 Average Contribution Criterion

3DGS decides densification from a Gaussian's positional gradients in screen space, averaged in magnitude across its observed views. For a Gaussian i observed in M views, let $g_{i,v} = \partial L/\partial \mu_i$ denote the gradient of the training loss L with respect to the Gaussian's projected screen-space mean μ_i in view v. A Gaussian is densified (*i.e.* split or cloned) when

$$\gamma_i^{\text{3DGS}} = \frac{1}{M} \sum_v \|g_{i,v}\| \tag{2}$$

exceeds a threshold. The denominator M weights every view equally: a view in which the Gaussian is barely visible counts the same as a fully visible one. As discussed in Sec. 3.1, fully occluded views inflate M while contributing nothing to the gradient sum, biasing the average downward.
To identify a more faithful measure of observation, we examine how the gradient is formed. The per-view gradient $g_{i,v} = (g_{i,v,x}, g_{i,v,y})$ is accumulated over pixels

to which Gaussian i contributes ($w_{i,p} > 0$), and the contribution of each pixel p is gated by the Gaussian's blending weight $w_{i,p}$. Per pixel, the gradient factorizes into the scalar $w_{i,p}$ and a weight-independent vector $s_{i,p}$:

$$g_{i,v} = \sum_p w_{i,p} s_{i,p}, \qquad w_{i,p} = \alpha_{i,p} T_{i,p}, \tag{3}$$

where the per-pixel vector $s_{i,p}$ combines the photometric residual, the geometric steering direction, and the local contrast of the Gaussian's color against what lies behind it, all independent of the Gaussian's visibility (detailed derivation in Sec. A.1). The blending weight $w_{i,p}$ is thus simultaneously the quantity with which the rasterizer composites the Gaussian's color and the factor that scales every term of its positional gradient. A pixel where the Gaussian is mostly occluded ($T_{i,p} \to 0$) or barely present ($\alpha_{i,p} \to 0$) contributes almost nothing to $g_{i,v}$, irrespective of that pixel's error. The gradient is thus already visibility-weighted at the pixel level, which motivates measuring a Gaussian's observation by its blending weights $w_{i,p}$ rather than by a binary view count.

We summarize a Gaussian's participation in view v by its mean blending weight $\bar{w}_{i,v}$ over the $N_{i,v}$ pixels in which it contributes as

$$\bar{w}_{i,v} = \frac{1}{N_{i,v}} \sum_p w_{i,p} \in [0, 1) \tag{4}$$

and replace the view count in Eq. (2) with the accumulated $\bar{w}_{i,v}$:

$$\gamma_i^{\text{Ours}} = \frac{1}{1 + \sum_v \bar{w}_{i,v}} \sum_v \|g_{i,v}\|. \tag{5}$$

We normalize by the mean per-view contribution $\bar{w}_{i,v}$ rather than the total $W_{i,v} = \sum_p w_{i,p} = N_{i,v}\bar{w}_{i,v}$. Any denominator that scales with the pixel count $N_{i,v}$ ties densification to screen-space footprint area and suppresses exactly the large Gaussians that are most likely to require it. In contrast, the mean measures how visible the Gaussian is where it is rendered, independently of coverage. Moreover, because $\bar{w}_{i,v}$ contains the transmittance $T_{i,p}$, the criterion is occlusion-aware without an explicit visibility test: a view in which the Gaussian is only faintly visible cannot dilute the average, unlike a view-count normalizer. Whether such a view pushes toward densification is then decided by its gradient alone: a faint view with a large residual contributes its full signal with minimal normalization cost, whereas a faint view with little error simply fades out.

This incidental occlusion-awareness is shared by the pixel-count normalizer of PixelGS [31]: a fully occluded view renders no pixels and drops out of its denominator as well. Beyond that, however, a visible-pixel count entangles view visibility with screen-space footprint, over-densifying near-camera Gaussians. PixelGS counteracts this with a separate distance scaling term based on the scene extent E. Our per-view mean $\bar{w}_{i,v}$ avoids this entanglement entirely. Additionally, through the transmittance $T_{i,p}$, it attenuates partial occlusion, grazing observations, and low-opacity views that leave a pixel count unchanged.

The constant in the denominator of Eq. (5) is a fixed floor: without it, rarely observed Gaussians ($\sum_v \bar{w}_{i,v} \to 0$) would receive unbounded scores where evidence is weakest. The floor ensures that scores increase smoothly and monotonically with accumulated evidence, placing rarely and frequently observed Gaussians on a common scale. A sweep in Sec. F.2 of the supplementary material confirms this: without the floor training diverges, while other floor values mainly re-scale the threshold. The updated criterion operates on a new scale and therefore requires a one-time recalibration of the densification threshold. A threshold sweep on a subset of the Mip-NeRF360 dataset shows 0.002 to be the operating point. We keep this value fixed across all experiments.
Finally, the normalization in Eq. (5) depends only on the per-pixel blending weights $w_{i,p}$, not on how the 2D gradients are reduced to a scalar value. Therefore, it applies unchanged to the absolute-gradient formulations of AbsGS [28] and GOF [29]. We apply the same criterion computation to all gradient terms.

4 Experiments

4.1 Datasets and Metrics

We evaluate on the common Gaussian Splatting scenes: Mip-NeRF360 [1] (9 scenes), Tanks & Temples [16] (2 scenes), and Deep Blending [11] (2 scenes). Additionally, we evaluate on Zip-NeRF [2], whose 4 large apartment scenes (alameda, berlin, london, nyc) exhibit heavy occlusions between rooms, with an occlusion rate of 0.63 against 0.15–0.48 on the other datasets (definition and per-scene values in Sec. D). Zip-NeRF is the established stress test for this regime: prior work reports 3DGS under-densification and over-smoothing on these scenes despite extensive tuning [5, 19]. For Zip-NeRF, we use images at half their original resolution, mirroring common 3DGS target resolutions, and hold out every 8th image as the validation set, following the standard protocol [13]. We report peak signal-to-noise ratio (PSNR), structural similarity (SSIM) [25], learned perceptual image patch similarity (LPIPS; VGG backbone) [30], the final number of Gaussians, and the training time. Large-scale scenes are prone to over-smoothing due to under-densification [5, 19]. We therefore also report the gradient magnitude similarity deviation (GMSD) [27], a metric specifically sensitive to blur. We read these metrics jointly: perceptual ones such as LPIPS credit reconstructed detail that *e.g.* PSNR penalizes when not pixel-aligned. Likewise, a lower Gaussian count is only better at comparable quality: a starved model is not more efficient.

4.2 Implementation

We embed our contributions in the integrated framework: the publicly available StopThePop [20], extended with the following improvements proposed by related work, which we do not claim as contributions. Unless stated otherwise, all runs use the opacity *decay* variant from StopThePop, and reported numbers are averaged over three training runs. All experiments run on an Nvidia A40 GPU.

Absolute gradient. We include the absolute-gradient condition of GOF [29], which improves high-frequency detail at an increased Gaussian count (Tab. 4).

Gaussian densification. We follow a split-only strategy, as also adopted by EFA-GS [23] and Deng *et al*. [4]. In our setting this removes a dependence on the scene extent E and avoids the excessive cloning that leads to pronounced floaters or total failure (Tab. 4). We also adopt Deng *et al*.'s geometrically meaningful long-axis split, which slightly reduces the count at competitive quality.

Pruning. Following the recommendation of Grubert *et al*. [9], we disable size-based pruning, which we find has no significant impact on quality or Gaussian count (Tab. 4). Disabling it also removes another dependence on the scene extent E. We additionally prune Gaussians whose maximum blending contribution w_{max} stays below a fixed threshold (0.005), as discussed by ReLaGS [26] and RadSplat [19]. We track w_{max} on the fly and prune throughout training [8, 21] on an automatic, hyperparameter-free schedule synchronized with the training queue. A single cheap pass over all training views at the end guarantees that the final model contains no fully occluded Gaussians (scheduling details in Sec. E.1, occluded-Gaussian counts per method in Sec. E.2).

Densification statistics. Original 3DGS accumulates densification statistics already during warm-up, where the largely transparent scene does not yet reflect its geometry. We exclude the warm-up period, which affects only the first densification event and is confirmed safe (Tab. 4).

Competitor comparison. We replicate PixelGS [31] and Grubert *et al*. [9] on the same StopThePop renderer in two settings: their full methods on vanilla StopThePop for the state-of-the-art comparison (Tab. 3), and their criterion alone inside the integrated framework, for the criterion comparison (Tab. 1).

4.3 Impact of Occlusion-Awareness

Integrating the adopted densification components into StopThePop already yields a strong but still occlusion-blind baseline. It surpasses StopThePop on all image-quality metrics on both datasets (Zip-NeRF: LPIPS by 6%, PSNR by 0.57 dB; StopThePop comparison in Sec. B). On top of this baseline, we isolate the occlusion-aware visibility filter (Sec. 3.1), leaving every other component unchanged. The top two rows of Tab. 1 compare the occlusion-blind baseline against its occlusion-aware counterpart. Making the filter occlusion-aware improves all image-quality metrics while enabling the densification that occlusion-blindness suppresses (LPIPS improves by a further 9%; the count rises from 0.56M to 2.57M). Merely lowering the occlusion-blind threshold is no equivalent control: it densifies indiscriminately rather than where occlusion suppressed it, and SMERF [5] reports that this produces floaters on these scenes. Occlusion-blind densification is thus a primary cause of under-densification on large, occlusion-heavy scenes. On Mip-NeRF360, where occlusions are far lighter,

Table 1: Occlusion-aware filter and densification criteria on the occlusion-heavy Zip-NeRF scenes (Mip-NeRF360 counterpart in Tab. S12). The occlusion-blind baseline, then varying only the criterion: the occlusion-aware observation count, the PixelGS pixel-count-weighted mean without and with distance scaling (+ DS), and our average contribution. Counts are in millions of Gaussians; time in minutes.

Method	PSNR↑	SSIM↑	LPIPS↓	GMSD↓	Count↓	Time↓
Occlusion-blind count	25.52	0.818	0.352	0.117	**0.557**	**22.3**
Occlusion-aware count	25.75	**0.834**	0.319	**0.103**	2.567	36.6
PixelGS	25.11	0.829	**0.312**	0.105	6.607	67.0
PixelGS + DS	25.59	0.824	0.341	0.112	1.443	28.8
Ours	**25.81**	0.830	0.330	0.106	1.133	29.9

the same pairing confirms the corrected filter does no harm: LPIPS improves by 4% at a moderately higher count (Sec. F.3).

4.4 Impact of the Average Contribution Criterion

We now compare densification criteria directly, holding the occlusion-aware filter and all other components fixed and varying only the criterion. We examine four criteria: the occlusion-aware observation count, the PixelGS [31] pixel-count-weighted mean without and with distance scaling (PixelGS, PixelGS + DS), and the average contribution (Ours). All criteria run at a single fixed threshold across every scene and dataset: the competitors at their published defaults, ours calibrated once on the Mip-NeRF360 sweep subset (Sec. F.1) and left unchanged elsewhere. Table 1 reports the four criteria on the occlusion-heavy Zip-NeRF scenes (Mip-NeRF360 counterpart in Tab. S12). On the smaller Mip-NeRF360 scenes the criterion barely affects quality (across the four criteria, PSNR spans 0.13 dB and LPIPS 3%), so compactness is decisive: our average-contribution criterion produces the fewest Gaussians (roughly 55–75% of the other criteria's counts) at matched quality. The criteria differ sharply on the large Zip-NeRF scenes. The raw pixel-count criterion is poorly behaved on its own: it over-densifies to 6.6M Gaussians, by far the largest count, yet reaches the worst PSNR. Under a long training schedule it diverges catastrophically (Sec. 4.5). PixelGS therefore couples the criterion with distance scaling as a guard rail against runaway densification. This bounds the count, but on these large multi-room scenes the guard suppresses densification that is actually needed: quality degrades to the worst SSIM, LPIPS, and GMSD of the four. We attribute this to the reliance of distance scaling on the scene extent. We find the scene extent to be unreliable in general but especially on large scenes (Sec. 4.2). Our method is instead fully independent of it. Our average-contribution criterion needs no guard rail: it is intrinsically self-limiting, giving the fewest Gaussians of the four at competitive quality. The occlusion-aware count is 0.5%, 3%, and 3% better on SSIM, LPIPS, and GMSD and 0.06 dB worse on PSNR, at 2.3× our count.

Table 2: Densification criteria on the Zip-NeRF scenes under longer schedules (100k and 210k iterations; single runs). Runs that crash from over-densification are marked as divergent. Counts are in millions of Gaussians; time in minutes.

	Method	PSNR↑	SSIM↑	LPIPS↓	GMSD↓	Count↓	Time↓
100k	Occlusion-aware count	25.90	0.842	**0.292**	0.095	4.792	178.5
	PixelGS	diverges (> 16.8M Gaussians, training crashes)					
	PixelGS + DS	26.35	0.843	0.302	0.094	2.843	129.5
	Ours	**26.48**	**0.847**	0.295	**0.091**	**1.890**	**105.6**
210k	Occlusion-aware count	diverges on 2 of 4 scenes (alameda, nyc)					
	PixelGS + DS	26.46	0.848	0.288	0.090	4.020	328.4
	Ours	**26.60**	**0.852**	**0.282**	**0.087**	**2.495**	**262.9**

This demand is structural: given more densification opportunity, it keeps growing until training diverges, whereas ours stays bounded (Sec. 4.5).

4.5 Scalability under a Matched Training Budget

The large Zip-NeRF scenes are unevenly observed: training cameras cluster in interior hotspots and cover peripheral regions far more sparsely (camera map for the alameda example in Fig. S3). They also hold about seven times as many training images as Mip-NeRF360, so the default 30k schedule grants each image only 22 optimization steps against Mip-NeRF360's 159, and lets it contribute to only 11 densification events against 77. Sparsely covered regions therefore receive few densification opportunities, and blur there can signal missing opportunity rather than poor densification decisions. Longer schedules restore this opportunity and reveal whether a criterion resolves the remaining detail or leaves these regions under-densified, and whether its growth stays bounded while doing so. We therefore additionally train the large Zip-NeRF scenes under a matched schedule of 210k iterations, which restores both quantities (157 steps and 76 densification events per image), and sample around it at 100k and 300k. Every other result keeps the standard 30k protocol for comparability. Table 2 shows the results up to the matched schedule; the 300k table and schedule details are in Sec. G.2.

Under these schedules the naive pixel-count criterion (PixelGS without distance scaling) is not merely inefficient but unstable: on every Zip-NeRF scene it over-densifies past the ∼16.8M trainable-count limit until training crashes (count-versus-iteration plot in Fig. S4). That limit lies nearly an order of magnitude beyond ours. The full PixelGS baseline, which includes the distance-scaling guard rail, does not diverge at its default threshold; the instability is a property of the pixel-count criterion in isolation. Our average-contribution criterion stays bounded throughout (2.75M even at 300k) and continues to improve in LPIPS and GMSD, whereas the guarded PixelGS remains behind on every image-quality metric while using 62% more Gaussians. Section G.4 traces this remaining gap to the guard itself: the distance term systematically suppresses near-object detail.

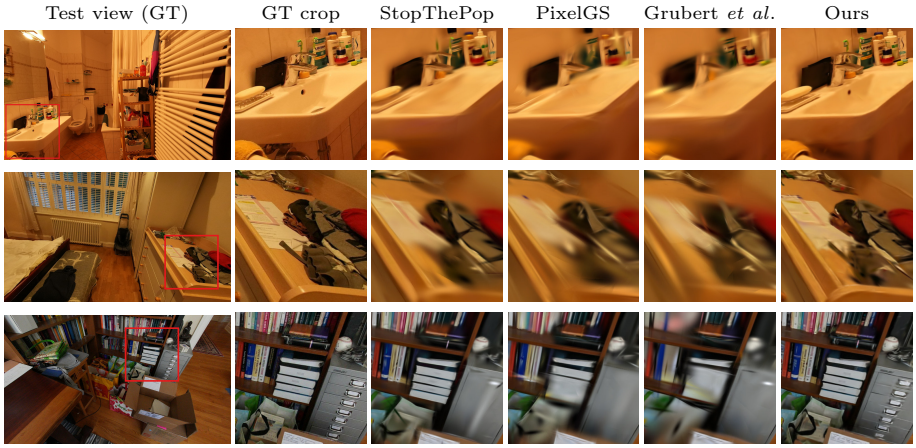


Fig. 3: Qualitative comparison on Zip-NeRF scenes (berlin, london, nyc; 30k schedule): StopThePop [20], our re-implementations of PixelGS [31] and Grubert *et al.* [9] on the StopThePop base, and our full model. The crops show the areas marked with red boxes in the test view. The state-of-the-art methods remain blurry in these occlusion-heavy scenes, while ours reconstructs fine geometry.

At 210k, even the occlusion-aware count over-densifies past the same limit on the two occlusion-heaviest scenes. These divergences are mechanistic, not single-run artifacts: the Gaussian count grows steadily into the trainable limit. We therefore also report the 100k schedule, the longest that the remaining count-based criteria complete, to quantify their high-count behavior. There, the extra Gaussians buy no quality advantage, repeating the Mip-NeRF360 pattern: ours leads PSNR, SSIM, and GMSD outright and stays within about 1% LPIPS of the occlusion-aware count, at 39–67% of the competing counts. Two supplementary analyses corroborate this. A threshold sweep (Sec. F.5) confirms that the self-limiting behavior is not specific to the chosen threshold: at half the default, the count-based criteria reach roughly three times our count. A per-view analysis (Sec. G.5) shows that our per-view quality stays as tight as the guarded PixelGS's, while the unguarded occlusion-aware count develops a growing low-quality tail.

4.6 Comparison to State-of-the-Art

Table 3 compares our method against StopThePop [20], our shared-base re-implementations of PixelGS and Grubert *et al.*, and published reference results. Across all four datasets our method reaches competitive or better image quality while using far fewer Gaussians than the strongest competitors, improving reconstruction efficiency rather than trading count for quality. On the occlusion-heavy Zip-NeRF scenes, both our variants lead every image-quality metric among the methods trained on the shared base, with up to 12% lower LPIPS than StopThePop and 8% (decay) to 11% (reset) better LPIPS than the best competitor. The

Table 3: Comparison on Deep Blending, Mip-NeRF360, Tanks & Temples and Zip-NeRF. We list results as reported by Grubert *et al.* [9] (†) and RadSplat [19] (‡). All other results are from our own experiments on the shared StopThePop (STP) base. *G* is Gaussians in millions; rows are grouped by base, with RadSplat and Zip-NeRF as upper-bound references (excluded from the color scale).

	Dataset	Deep Blending				Mip-NeRF360				Tanks & Temples				Zip-NeRF			
	Metric	PSNR↑	SSIM↑	LPIPS↓	G(M)↓	PSNR↑	SSIM↑	LPIPS↓	G(M)↓	PSNR↑	SSIM↑	LPIPS↓	G(M)↓	PSNR↑	SSIM↑	LPIPS↓	G(M)↓
3DGS	PixelGS†	28.95	0.892	0.250	∼4.6	27.53	0.822	0.191	∼5.5	23.83	0.854	0.151	∼4.5	N/A	N/A	N/A	N/A
	Grubert *et al.* †	29.63	0.902	0.240	∼2.3	**27.71**	**0.824**	0.193	∼3.3	**24.18**	**0.860**	0.149	∼2.4	N/A	N/A	N/A	N/A
StopThePop	STP (Reset)	29.91	0.904	0.234	2.819	27.30	0.814	0.214	3.289	23.15	0.843	0.173	1.800	25.11	0.808	0.366	0.951
	PixelGS	29.54	0.901	0.234	4.392	27.37	0.822	0.190	5.328	23.27	0.852	0.147	4.175	25.15	0.811	0.360	1.980
	Grubert *et al.*	**30.05**	**0.905**	0.232	2.209	27.59	**0.824**	0.192	3.263	23.78	0.858	0.146	2.231	25.27	0.812	0.360	1.131
	Ours (Reset)	29.90	0.902	**0.231**	1.656	27.42	0.817	**0.184**	3.913	23.16	0.848	0.140	3.871	**25.86**	**0.834**	**0.320**	1.991
	STP (Decay)	29.80	0.904	0.242	1.219	27.01	0.809	0.227	**1.711**	23.09	0.839	0.185	**0.848**	24.95	0.802	0.376	**0.486**
	Ours (Decay)	30.03	**0.905**	0.234	**0.894**	27.30	0.820	0.186	1.954	23.22	0.853	**0.137**	1.965	25.81	0.830	0.330	1.133
NeRF	RadSplat‡	N/A	N/A	N/A	N/A	28.14	0.843	0.171	1.924	N/A	N/A	N/A	N/A	26.17	0.839	0.364	N/A
	Zip-NeRF‡	N/A	N/A	N/A	-	28.54	0.836	0.177	-	N/A	N/A	N/A	-	27.37	0.836	0.305	-

decay variant achieves this at the same Gaussian count as Grubert *et al.*, the strongest competitor. Figure 3 shows this qualitatively: in heavily occluded interior regions, the occlusion-blind methods remain visibly blurry, while our method resolves the fine geometry. Even the RadSplat upper-bound reference trails both our variants in LPIPS, by 9–12%. On Deep Blending it matches the best quality (within 0.02 dB PSNR, tied SSIM with Grubert *et al.*) at less than half the Gaussians. On Mip-NeRF360 it achieves better LPIPS than every StopThePop, PixelGS, and Grubert variant with its other metrics within about 2% of the best. The decay variant uses 35–60% of the PixelGS and Grubert counts. On Tanks & Temples the competitors' paper-reported PSNR advantage shrinks substantially on the shared base (Grubert *et al.* remain 0.56 dB ahead). Ours still holds the best LPIPS and stays within roughly 1% on SSIM, the pattern expected when detail is reconstructed but not perfectly pixel-aligned. Across scenes, the occlusion rate predicts where our contributions help: it correlates with our LPIPS gain over the occlusion-blind variant (Spearman +0.83). Ours also allocates its Gaussians more efficiently: even without the contribution prune, our models contain the smallest share of fully occluded Gaussians (7.2% on Zip-NeRF, against 10.5–22.7% for the occlusion-blind methods; occluded-Gaussian counts per method in Tab. S8). This is no artifact of lower counts: at a nearly matched count, our unpruned variant contains 43% fewer occluded Gaussians than Grubert *et al.*'s.

4.7 Ablation Study

We ablate our full model on the Zip-NeRF scenes by removing or reverting one component at a time (Tab. 4). The isolation of our two contributions on vanilla StopThePop, and the clone failure it exposes, are analyzed in Sec. C.2 (Tab. S5) of the supplementary material. By far the most consequential component is split-only densification: restoring the original clone operation (*+ clone*) collapses reconstruction quality (nearly 5 dB PSNR and 31% LPIPS worse). Cloning depends on the unreliable scene extent that these large scenes expose. The GOF absolute-gradient signal (*- GOF Grad*) is the largest quality contribu-

Table 4: Ablation on the Zip-NeRF scenes: removing (−) or adding (+) one component relative to our full model. The catastrophic failure row (gray) is excluded from the color scale. Counts are in millions of Gaussians; time in minutes.

Components	PSNR↑	SSIM↑	LPIPS↓	GMSD↓	Count↓	Time↓
Full model	**25.81**	0.830	0.330	0.106	1.133	29.9
+ clone	20.86	0.735	0.432	0.175	0.760	28.6
- GOF Grad	25.56	0.818	0.351	0.119	**0.781**	**23.8**
- long-axis split	25.80	**0.833**	**0.321**	**0.103**	1.635	29.6
- contrib prune	25.79	0.830	0.330	0.106	1.245	27.9
+ warm-up stats	**25.81**	0.830	0.330	0.106	1.134	26.7
+ size prune	25.80	0.830	0.330	0.106	1.133	26.6

tor among the remaining components (6% better LPIPS at a moderately higher count). Long-axis split reduces the count by 31% at a marginal 3% LPIPS trade, contribution pruning by 9% at unchanged quality, while the statistics warm-up and size prune are negligible. Reverting opacity decay to reset gives marginally better quality at 76% more Gaussians (both variants in Tab. 3), confirming decay as the more efficient default.

5 Conclusion & Limitations

We expose the occlusion-blindness of the adaptive density control in standard 3DGS and propose a modular solution, transferable to existing Gaussian Splatting methods, that ensures occlusion-aware densification statistics. We further present a novel densification criterion that normalizes a Gaussian's accumulated screen-space gradients by the accumulated average contributions of its rendered views. This avoids the view- and pixel-count biases of previous criteria under occlusion, large footprints, and varying distance. This leads to perceptually better reconstructions at fewer Gaussians on large, occlusion-heavy scenes. We additionally integrate established densification improvements into a framework that reduces the dependence on unreliable heuristics such as the scene extent.
Large-scale reconstruction nevertheless does not yet reach small-scene quality. Beyond occlusion, the imbalanced camera coverage of the Zip-NeRF [2] scenes limits what densification can recover (Sec. 4.5). Longer schedules restore the missing densification opportunity, but a fairer distribution of observations would address the imbalance at its source. Several works group cameras by scene sections to pursue divide-and-conquer strategies [14,17]. Combining occlusion-aware densification with a balanced training strategy should enable higher-quality large-scale reconstructions via single joint optimizations. Our code will be made available at `https://github.com/MarcelRogge/occlusion-aware-3dgs`.

Acknowledgement. This work was partially funded by the European Union under the grant number 101298864, project Victor-XR.

References

1. Barron, J.T., Mildenhall, B., Verbin, D., Srinivasan, P.P., Hedman, P.: Mip-NeRF 360: Unbounded anti-aliased neural radiance fields. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). pp. 5460–5469 (2022)
2. Barron, J.T., Mildenhall, B., Verbin, D., Srinivasan, P.P., Hedman, P.: Zip-NeRF: Anti-aliased grid-based neural radiance fields. In: Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV). pp. 19640–19648 (2023)
3. Cheng, K., Long, X., Yang, K., Yao, Y., Yin, W., Ma, Y., Wang, W., Chen, X.: GaussianPro: 3D Gaussian splatting with progressive propagation. In: Proceedings of the International Conference on Machine Learning (ICML) (2024)
4. Deng, X., Diao, C., Li, M., Yu, R., Xu, D.: Improving densification in 3D Gaussian splatting for high-fidelity rendering (2025), https://arxiv.org/abs/2508.12313
5. Duckworth, D., Hedman, P., Reiser, C., Zhizhin, P., Thibert, J.F., Lučić, M., Szeliski, R., Barron, J.T.: SMERF: Streamable memory efficient radiance fields for real-time large-scene exploration. ACM Transactions on Graphics **43**(4), 1–13 (2024)
6. Fang, G., Wang, B.: Mini-Splatting: Representing scenes with a constrained number of Gaussians. In: Proceedings of the European Conference on Computer Vision (ECCV). pp. 165–181 (2024). https://doi.org/10.1007/978-3-031-72980-5_10
7. Fridovich-Keil, S., Yu, A., Tancik, M., Chen, Q., Recht, B., Kanazawa, A.: Plenoxels: Radiance fields without neural networks. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) (2022)
8. Girish, S., Gupta, K., Shrivastava, A.: EAGLES: Efficient accelerated 3D Gaussians with lightweight encodings. In: Proceedings of the European Conference on Computer Vision (ECCV). pp. 54–71 (2024). https://doi.org/10.1007/978-3-031-73036-8_4
9. Grubert, G., Barthel, F., Hilsmann, A., Eisert, P.: Improving adaptive density control for 3D Gaussian splatting. In: Proceedings of the 20th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications. pp. 610–621 (2025). https://doi.org/10.5220/0013308500003912
10. Hahlbohm, F., Friederichs, F., Weyrich, T., Franke, L., Kappel, M., Castillo, S., Stamminger, M., Eisemann, M., Magnor, M.: Efficient perspective-correct 3D Gaussian splatting using hybrid transparency. Computer Graphics Forum **44**(2) (2025). https://doi.org/10.1111/cgf.70014, https://fhahlbohm.github.io/htgs/
11. Hedman, P., Philip, J., Price, T., Frahm, J.M., Drettakis, G., Brostow, G.: Deep Blending for free-viewpoint image-based rendering. ACM Transactions on Graphics **37**(6) (Dec 2018). https://doi.org/10.1145/3272127.3275084
12. Huang, B., Yu, Z., Chen, A., Geiger, A., Gao, S.: 2D Gaussian splatting for geometrically accurate radiance fields. In: SIGGRAPH 2024 Conference Papers (2024). https://doi.org/10.1145/3641519.3657428
13. Kerbl, B., Kopanas, G., Leimkühler, T., Drettakis, G.: 3D Gaussian splatting for real-time radiance field rendering. ACM Transactions on Graphics **42**(4) (July 2023), https://repo-sam.inria.fr/fungraph/3d-gaussian-splatting/
14. Kerbl, B., Meuleman, A., Kopanas, G., Wimmer, M., Lanvin, A., Drettakis, G.: A hierarchical 3D Gaussian representation for real-time rendering of very large datasets. ACM Transactions on Graphics **43**(4) (July 2024), https://repo-sam.inria.fr/fungraph/hierarchical-3d-gaussians/

15. Kheradmand, S., Rebain, D., Sharma, G., Sun, W., Tseng, Y.C., Isack, H., Kar, A., Tagliasacchi, A., Yi, K.M.: 3D Gaussian splatting as Markov chain Monte Carlo. In: Advances in Neural Information Processing Systems (NeurIPS). vol. 37, pp. 80965–80986 (2024)
16. Knapitsch, A., Park, J., Zhou, Q.Y., Koltun, V.: Tanks and Temples: Benchmarking large-scale scene reconstruction. ACM Transactions on Graphics **36**(4) (2017)
17. Liu, S., Tang, X., Li, Z., He, Y., Ye, C., Liu, J., Huang, B., Zhou, S., Wu, X.: OccluGaussian: Occlusion-aware Gaussian splatting for large scene reconstruction and rendering. In: Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV). pp. 26643–26652 (October 2025)
18. Mallick, S.S., Goel, R., Kerbl, B., Steinberger, M., Carrasco, F.V., De La Torre, F.: Taming 3DGS: High-quality radiance fields with limited resources. In: SIGGRAPH Asia 2024 Conference Papers (2024). https://doi.org/10.1145/3680528.3687694
19. Niemeyer, M., Manhardt, F., Rakotosaona, M.J., Oechsle, M., Duckworth, D., Gosula, R., Tateno, K., Bates, J., Kaeser, D., Tombari, F.: RadSplat: Radiance field-informed Gaussian splatting for robust real-time rendering with 900+ FPS. In: International Conference on 3D Vision (3DV) (2025)
20. Radl, L., Steiner, M., Parger, M., Weinrauch, A., Kerbl, B., Steinberger, M.: StopThePop: Sorted Gaussian splatting for view-consistent real-time rendering. ACM Transactions on Graphics **43**(4) (2024)
21. Rogge, M., Stricker, D.: Object-centric 2D Gaussian splatting: Background removal and occlusion-aware pruning for compact object models. In: Proceedings of the 14th International Conference on Pattern Recognition Applications and Methods - ICPRAM. pp. 519–530 (2025). https://doi.org/10.5220/0013305500003905
22. Rota Bulò, S., Porzi, L., Kontschieder, P.: Revising densification in Gaussian splatting. In: Proceedings of the European Conference on Computer Vision (ECCV). pp. 347–362 (2024). https://doi.org/10.1007/978-3-031-73036-8_20
23. Wang, J., Zhou, P., Li, C., Quan, R., Qin, J.: Low-frequency first: Eliminating floating artifacts in 3D Gaussian splatting. In: International Conference on Cyberworlds (CW). pp. 1–8 (2025)
24. Wang, P., Wang, Y., Wang, D., Mohan, S., Fan, Z., Wu, L., Cai, R., Yeh, Y.Y., Wang, Z., Liu, Q., Ranjan, R.: Steepest descent density control for compact 3D Gaussian splatting. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) (2025)
25. Wang, Z., Bovik, A.C., Sheikh, H.R., Simoncelli, E.P.: Image quality assessment: From error visibility to structural similarity. IEEE Transactions on Image Processing **13**(4), 600–612 (2004)
26. Xie, Y., Arafa, A., Javanmardi, A., Millerdurai, C., Hu, J.C., Wang, S., Pagani, A., Stricker, D.: ReLaGS: Relational language Gaussian splatting. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) (2026)
27. Xue, W., Zhang, L., Mou, X., Bovik, A.C.: Gradient magnitude similarity deviation: A highly efficient perceptual image quality index. IEEE Transactions on Image Processing **23**(2), 684–695 (2014). https://doi.org/10.1109/TIP.2013.2293423
28. Ye, Z., Li, W., Liu, S., Qiao, P., Dou, Y.: AbsGS: Recovering fine details in 3D Gaussian splatting. In: Proceedings of the ACM International Conference on Multimedia (ACM MM). pp. 1053–1061 (2024). https://doi.org/10.1145/3664647.3681361

29. Yu, Z., Sattler, T., Geiger, A.: Gaussian opacity fields: Efficient adaptive surface reconstruction in unbounded scenes. ACM Transactions on Graphics **43**(6), 1–13 (2024)
30. Zhang, R., Isola, P., Efros, A.A., Shechtman, E., Wang, O.: The unreasonable effectiveness of deep features as a perceptual metric. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) (2018)
31. Zhang, Z., Hu, W., Lao, Y., He, T., Zhao, H.: Pixel-GS: Density control with pixel-aware gradient for 3D Gaussian splatting. In: Proceedings of the European Conference on Computer Vision (ECCV) (2024)
32. Zwicker, M., Pfister, H., van Baar, J., Gross, M.: EWA volume splatting. In: Proceedings of the Conference on Visualization '01. pp. 29–36 (2001)

Supplementary Material

This document supplements the main paper. References to the main paper's figures, tables, and equations are written out explicitly, while all other references resolve within this document. As in the main paper, all runs use the (Decay) opacity schedule unless stated otherwise, and evaluation follows the protocol of the main paper's experiments section.

Overview. Section A derives the densification gradient behind our average-contribution criterion and transfers both contributions to 2DGS [12] and 3DGS [13]. Section B quantifies the integrated baseline against vanilla StopThePop. Section C analyzes the clone failure mode behind our split-only decision, including the scene-extent override and an ablation study. Section D quantifies how occluded each benchmark is and shows that the per-scene occlusion rate predicts where our contributions help. Section E details the contribution-prune schedule and takes a census of fully occluded Gaussians across methods. Section F compares densification criteria on Mip-NeRF360, maps where each allocates Gaussians, and examines the denominator floor, robustness to the threshold and the evaluation resolution, a hybrid of PixelGS and our criterion, and two alternative normalizers. Section G covers extended training schedules, count growth over iterations, the distance-scaling guard rail, and per-view statistics. Section H closes with full-frame qualitative results and error maps.

A Method Details

In this section, we derive our average-contribution criterion from the per-pixel blending weights (Sec. A.1) and test how far our two contributions generalize beyond StopThePop [20] by porting them to 2DGS [12] and 3DGS [13] (Sec. A.2).

A.1 Criterion Derivation

We recall the projected footprint of Gaussian i at pixel p,

$$G_{i,p} = \exp\big(-\tfrac{1}{2}\,(p-\mu_i)^T\,\Sigma_i'^{-1}\,(p-\mu_i)\big), \tag{S1}$$

with the projected screen-space mean μ_i and the projected 2D covariance Σ_i'. As in the main text, the densification gradient differentiates with respect to this projected mean, so all position derivatives below are two-dimensional.
The 2D screen-space gradient $g_{i,v} = (g_{i,v,x}, g_{i,v,y})$ is used to determine how much a Gaussian i wants to move based on the loss computed in view v. The same holds independently for every Gaussian. Kerbl *et al.* [13] argue that a Gaussian that tries to fill many places at once requires densification. Therefore, the magnitude $\|g_{i,v}\|$ is used for the criterion to determine densification. For a single fixed view v, we drop the view index and write $g_i \equiv g_{i,v}$. Additionally, we consider only the

derivation of $g_{i,x}$, since $g_{i,y}$ follows analogously.

$$g_{i,x} = \frac{\partial L}{\partial \mu_{i,x}} = \sum_p \frac{\partial L_p}{\partial \mu_{i,x}} = \sum_p \frac{\partial L_p}{\partial \alpha_{i,p}} \cdot \frac{\partial \alpha_{i,p}}{\partial \mu_{i,x}} = \sum_p \frac{\partial L_p}{\partial C_p} \cdot \frac{\partial C_p}{\partial \alpha_{i,p}} \cdot \frac{\partial \alpha_{i,p}}{\partial \mu_{i,x}}, \quad \text{(S2)}$$

where $\sum_p$ runs over the N_i pixels to which Gaussian i contributes in view v. Using the chain rule, we express the gradient's per-pixel term as a product of three derivatives: $\partial L_p/\partial C_p$ represents the photometric error, $\partial C_p/\partial \alpha_{i,p}$ represents the blending sensitivity, and $\partial \alpha_{i,p}/\partial \mu_{i,x}$ represents the positional sensitivity.

$$\frac{\partial C_p}{\partial \alpha_{i,p}} = (c_i \cdot T_{i,p} - A^{\text{gs}}_{i,p} \cdot T_{i,p} - c_{bg} \cdot T^{\text{tail}}_{i,p} \cdot T_{i,p}) = (c_i - A_{i,p}) \cdot T_{i,p}, \quad \text{(S3)}$$

where $A^{\text{gs}}_{i,p}$ represents the color composited from all Gaussians behind Gaussian i, c_{bg} is the background fill color, $A_{i,p}$ expresses the entire color composite behind Gaussian i, $T_{i,p}$ is the transmittance reaching Gaussian i, and $T^{\text{tail}}_{i,p}$ represents the transmittance behind Gaussian i. Note that $T_{i,p} \cdot (1 - \alpha_{i,p}) \cdot T^{\text{tail}}_{i,p} = T_{final,p}$, where $T_{final,p}$ represents the leftover transmittance after blending all Gaussians overlapping pixel p. We can see that the blending sensitivity consists of the transmittance $T_{i,p}$ at Gaussian i and a contrast term between the color c_i of Gaussian i and the composite color $A_{i,p}$ behind it.

$$\frac{\partial \alpha_{i,p}}{\partial \mu_{i,x}} = o_i \cdot \frac{\partial G_{i,p}}{\partial \mu_{i,x}} = o_i \cdot G_{i,p} \cdot \frac{\partial \ln(G_{i,p})}{\partial \mu_{i,x}} = \alpha_{i,p} \cdot \frac{\partial \ln(G_{i,p})}{\partial \mu_{i,x}}, \quad \text{(S4)}$$

where o_i is the peak opacity of Gaussian i (at its center) and $\alpha_{i,p} = o_i G_{i,p}$ is its opacity at pixel p, attenuated by the Gaussian's footprint $G_{i,p} \in (0, 1]$ on pixel p. The positional sensitivity term consists of a geometric steering term $\partial \ln(G_{i,p})/\partial \mu_{i,x}$ modulated by the Gaussian's opacity at pixel p.
Inserting Eq. (S3) and Eq. (S4) into Eq. (S2) yields:

$$\begin{aligned} g_{i,x} &= \sum_p \frac{\partial L_p}{\partial C_p} \cdot (c_i - A_{i,p}) \cdot T_{i,p} \cdot \alpha_{i,p} \cdot \frac{\partial \ln(G_{i,p})}{\partial \mu_{i,x}} \\ &= \sum_p \frac{\partial L_p}{\partial C_p} \cdot (c_i - A_{i,p}) \cdot w_{i,p} \cdot \frac{\partial \ln(G_{i,p})}{\partial \mu_{i,x}}, \end{aligned} \quad \text{(S5)}$$

where $w_{i,p} = \alpha_{i,p} T_{i,p}$ is the Gaussian's weight with which its color is blended into pixel p, *i.e.* its contribution. Equation (S5) shows that $g_{i,x}$ consists of a photometric error, a geometric steering, and a contrast term, which are all scaled according to its contribution $w_{i,p}$. These three weight-independent factors form the per-pixel vector $s_{i,p}$ of Eq. (3) in the main paper. This shows that the gradient is intrinsically contribution-weighted, which motivates our normalization using the mean contribution $\bar{w}_i = (1/N_i) \sum_p w_{i,p}$ rather than a raw view count. The full criterion accumulates these per-view means over all views and adds the $+1$ floor (Eq. (5) of the main paper), which bounds the score of a Gaussian that has accumulated little contribution (more details on the floor in Sec. F.2). A contribution-based denominator is therefore the matched choice, though not

every contribution-based denominator will do: the per-view *total* $\sum_p w_{i,p}$ grows with the on-screen footprint and would reintroduce the very pixel-count dependence we argue against (Sec. 3.2 of the main paper), whereas the per-view mean $\bar{w}_i$ is footprint-free.

A.2 Generalizability (2DGS & 3DGS)

Our contributions rely only on the per-pixel blending weights, so they should transfer to most other Gaussian Splatting renderers. To test this, we port both contributions to 2DGS [12], whose surfel-based rasterizer and regularization differ substantially from StopThePop [20]. We additionally port them to 3DGS [13], which shares StopThePop's splatting formulation but none of its per-pixel sorting. We do not include any of the other adopted components to isolate the generality of our proposed contributions. We report results on each vanilla base, the occlusion-aware filter alone (*+ filter*), and the filter with criterion (*+ filter + criterion*).

Mip-NeRF360. We calibrate each port with a small threshold sweep on the subset of the Mip-NeRF360 [1] dataset used for our own calibration (Sec. F.1), reported in Tab. S1. Both ports behave the same way. At our original threshold of 0.002, the criterion improves every quality metric over both the baseline and the filter-only variant, and on both ports it does so at a markedly increased Gaussian count (1.8× on 2DGS, 2.4× on 3DGS). Raising the threshold to 0.003 returns the count closer to the baseline's level on both ports while retaining much of the gain. Raising it further under-densifies, with every metric falling below the baseline on both 2DGS and 3DGS. The port's missing contribution prune does not explain the inflated count at our default threshold of 0.002: disabling it in our StopThePop framework raises the Mip-NeRF360 count by only 4% (Tab. S8). Later in this section, we will discuss a possible explanation in the clone operation.

Zip-NeRF. Table S2 reports both ports on the occlusion-heavy Zip-NeRF [2] scenes. For 2DGS we find that 0.003 does not allow much more densification than the base, so we report the results for a threshold of 0.002. For 3DGS we report with the recalibrated threshold of 0.003. In both cases, we avoid the clone's catastrophic failure through a manual override of the scene extent (Sec. C.1) so that the original densification strategy can be retained. Both baselines are starved here, at 0.57M and 1.10M Gaussians against the 1.8–2.8M they use on the smaller Mip-NeRF360 scenes, which is the same under-densification our contributions correct on StopThePop. The filter alone lifts the count sharply on both, to 3.03M and 2.90M, and the criterion then spends that budget more selectively, settling at 1.48M and 1.68M. On 2DGS the full port improves every metric over the baseline on three of the four scenes (+0.38 to +0.58 dB, 5–8% better LPIPS). On 3DGS it attains the best PSNR on all four scenes, while conceding a little SSIM and LPIPS to the filter-only variant. For 2DGS there is an exception on alameda, where it degrades: the vanilla 2DGS already

Table S1: Both ports on the Mip-NeRF360 subset of the threshold sweep (single runs, averaged over the four scenes): the vanilla base, the occlusion-aware filter alone, and the full port across the densification-threshold sweep. Shading compares within each base. Counts are in millions of Gaussians.

	Method	PSNR↑	SSIM↑	LPIPS↓	Count↓
2DGS	2DGS	27.69	0.850	0.218	2.059
	+ filter	27.87	0.857	0.204	2.512
	+ filter + criterion (0.001)	over-densifies; out of memory			
	+ filter + criterion (0.002)	**28.04**	**0.865**	**0.182**	3.682
	+ filter + criterion (0.003)	27.89	0.855	0.204	2.184
	+ filter + criterion (0.004)	27.67	0.845	0.225	**0.910**
3DGS	3DGS	28.46	0.868	0.181	2.770
	+ filter	28.48	0.871	0.176	3.126
	+ filter + criterion (0.002)	**28.71**	**0.877**	**0.158**	6.519
	+ filter + criterion (0.003)	28.65	0.873	0.171	3.347
	+ filter + criterion (0.004)	28.46	0.865	0.187	1.990
	+ filter + criterion (0.005)	28.25	0.854	0.207	**1.302**

exhibits large floater artifacts that we attribute to this scene's strong exposure and lighting changes, which the surfel geometry consolidates into large floater blobs. Due to densification no longer getting suppressed, these floater clusters are amplified. Our criterion reduces the number of floaters and recovers most of the lost quality, while the 3DGS port does not show this degradation as it does not form the floaters into solid clusters.

The clone operation. To investigate the inflated counts for our criterion on the Mip-NeRF360 scenes, we isolate the effect of the clone operation, which we already identify as being responsible for a catastrophic failure on large-scale scenes in Sec. C. We disable cloning by overriding the scene extent (Sec. C.1) such that every densification becomes a split, and compare against an otherwise identical run with cloning enabled. Notably, the size prune is disabled in both cases since the scene-extent override would otherwise result in a different behavior between the clone and split-only runs. Table S3 shows the results for both 2DGS and 3DGS. The result agrees across both ports: removing the clone discards between 18% and 43% of the final Gaussians while changing PSNR by at most 0.25 dB and LPIPS by at most 0.006. On 2DGS at our default threshold of 0.002, it cuts the count by 43%. This reduces the final count below the level of the baseline while producing better quality metrics, which mirrors the behavior of the criterion that we find on the StopThePop base. However, for 3DGS this alone does not restore the expected behavior on the default threshold of 0.002. We see that cloning accounts for a significant amount of the final Gaussians without notable quality improvements. This is not the catastrophic failure it produces on the large scenes, but it shows the same mechanism: cloning causes over-densification if not suppressed.

Table S2: Both ports on the Zip-NeRF scenes (30k, single runs), at each port's own densification threshold (0.002 for 2DGS, 0.003 for 3DGS). Shading compares within each scene and base. Counts are in millions of Gaussians.

		2DGS				3DGS			
	Method	PSNR↑	SSIM↑	LPIPS↓	Count↓	PSNR↑	SSIM↑	LPIPS↓	Count↓
alameda	Vanilla	**20.90**	**0.689**	**0.456**	**0.630**	22.28	0.728	0.400	**1.216**
	+ filter	18.95	0.661	0.480	3.222	22.40	**0.740**	**0.380**	2.654
	+ filter + crit.	20.13	0.684	0.457	1.443	**22.46**	0.736	0.388	1.806
berlin	Vanilla	25.79	0.865	0.359	**0.556**	27.10	0.886	0.326	**1.024**
	+ filter	25.48	0.870	0.343	3.039	27.39	**0.895**	**0.308**	3.142
	+ filter + crit.	**26.17**	**0.875**	**0.341**	1.587	**27.44**	0.891	0.316	1.710
london	Vanilla	24.33	0.768	0.428	**0.414**	25.52	0.804	0.380	**0.835**
	+ filter	24.20	0.783	**0.396**	2.253	25.83	**0.816**	**0.358**	1.870
	+ filter + crit.	**24.73**	**0.787**	0.397	1.136	**25.83**	0.813	0.365	1.253
nyc	Vanilla	25.70	0.815	0.368	**0.674**	26.76	0.840	0.328	**1.335**
	+ filter	24.11	0.809	0.356	3.612	26.97	**0.851**	**0.304**	3.948
	+ filter + crit.	**26.28**	**0.833**	**0.339**	1.734	**27.08**	0.847	0.315	1.937

B The Integrated Baseline

We present results of our integrated baseline in Tab. S4. Adopting only the components from related work described in Sec. 4.2 of the main paper, without our proposed contributions, already results in an improvement across all quality metrics. The Gaussian count increases in all cases, which could be interpreted as worse. However, we believe the baseline still under-densifies on Zip-NeRF: the count is lower than on the Mip-NeRF360 scenes even though the apartments are larger and more detailed. For Mip-NeRF360, it is more difficult to evaluate which count is appropriate. Even so, we consider the 14% LPIPS improvement a meaningful gain that justifies the 24% increase in count.

C Clone Analysis

In this section, we give a small analysis of the default clone operation from 3DGS (Sec. 2.1 of the main paper). Specifically, we notice the creation of large floater clusters when the scene extent E is too large relative to the detail in the scene. The significant impact of this is also visible in the main paper's ablation, where adding clone to our proposed method causes catastrophic failure (Sec. 4.7 of the main paper).

C.1 Manual Scene-Extent Override

The scene extent E is automatically computed based on the camera positions and does not actually identify what size is appropriate for fine detail. We see this

Table S3: Isolating the clone operation on both ports, on the Mip-NeRF360 subset (single runs, averaged over the four scenes). Shading compares within each base. Counts are in millions of Gaussians.

	Threshold	Densification	PSNR↑	SSIM↑	LPIPS↓	Count↓
2DGS	0.002	with clone	28.14	**0.867**	**0.181**	2.947
		split-only	**28.19**	0.865	0.185	1.696
	0.003	with clone	28.02	0.859	0.202	1.441
		split-only	27.97	0.855	0.206	**0.980**
3DGS	0.002	with clone	**28.85**	**0.878**	**0.157**	6.455
		split-only	28.60	0.870	0.164	5.267
	0.003	with clone	28.73	0.874	0.170	3.304
		split-only	28.64	0.869	0.174	**2.263**

Table S4: StopThePop [20] versus our integrated baseline (StopThePop with the adopted densification components, still occlusion-blind) on the Zip-NeRF (top) and Mip-NeRF360 (bottom) scenes (3-run averages). Counts are in millions of Gaussians; time in minutes.

	Method	PSNR↑	SSIM↑	LPIPS↓	GMSD↓	Count↓	Time↓
Zip	StopThePop	24.95	0.802	0.376	0.133	**0.486**	24.3
	Baseline	**25.52**	**0.818**	**0.352**	**0.117**	0.557	**22.3**
Mip	StopThePop	27.01	0.809	0.227	0.102	**1.711**	**25.6**
	Baseline	**27.29**	**0.821**	**0.196**	**0.087**	2.117	31.4

in the Zip-NeRF [2] apartments: cameras span multiple rooms (*e.g.* alameda in Fig. S3) while detail represents small areas within individual rooms. To isolate the effect of the clone operation based on the scene extent, we determine appropriate scene extent values for the Zip-NeRF scenes and set them manually for an ablation experiment. Specifically, we subdivide each scene from Zip-NeRF into smaller occlusion-light subsets such that we would expect an optimization on each subset to produce a high-quality reconstruction, as on the Mip-NeRF360 dataset. Per scene, we then compute the scene extent for each of these subsets and pick the smallest as the scene extent for the full scene. The original scene extents according to 3DGS' automatic estimation are as follows: 7.80 for alameda, 6.64 for berlin, 7.27 for london, and 7.52 for nyc. We override these with the following: 2.47 for alameda, 2.81 for berlin, 2.16 for london, and 1.25 for nyc. Accordingly, the extent-dependent ADC mechanisms (the split-versus-clone decision and the world-space size prune) operate on this manually corrected scale. Table S5 shows this manual override of the scene extent as *+ extent*, which will be discussed in more detail in Sec. C.2.

Grubert *et al.* [9] propose an alternative scene extent, computed as the mean distance from the average camera position to the SfM points. This can result in a smaller scene extent than the camera-based one if the cameras are well

Table S5: Complete clone-fix ablation on the Zip-NeRF scenes (3-run averages): StopThePop plus our two contributions and/or one clone fix—split-only densification or the manual scene-extent override of Sec. C.1. The catastrophic failure row (gray) is excluded from the color scale. Counts are in millions of Gaussians; time in minutes.

Components	PSNR↑	SSIM↑	LPIPS↓	GMSD↓	Count↓	Time↓
StopThePop	24.95	0.802	0.376	0.133	**0.486**	24.3
+ contributions	23.03	0.776	0.393	0.148	1.099	26.6
+ extent	25.06	0.809	0.365	0.127	0.564	20.1
+ contributions + extent	25.55	0.824	0.341	0.114	1.303	27.0
+ split-only	25.19	0.808	0.366	0.129	0.570	**19.7**
+ contributions + split-only	25.63	0.822	0.343	0.115	1.212	25.2
Full model	**25.81**	**0.830**	**0.330**	**0.106**	1.133	29.9

distributed throughout the scene, and a larger one if the cameras are restricted to a small area of the scene. On the Zip-NeRF apartments, it evaluates to 3.8–4.9, below the camera-based 6.6–7.8 but still well above the detail-appropriate values we derived above. Ultimately, the goal of such an estimate is to match the perceived size of the scene better than the camera-based extent. However, any such extent only reasons about the size of the full scene and not what size of a Gaussian is appropriate for detail within that scene. That is, a fixed detail size (*e.g.* a pen) is a larger fraction of a small scene's extent than of a large one; tying the Gaussian scale to scene extent therefore assumes a proportionality between detail and scene size that does not hold.

C.2 Clone Ablation

Table S5 shows a more detailed ablation of the clone operation. We evaluate two approaches to prevent the catastrophic failure that the clone operation causes on large scenes: a manual override of the scene *extent* as discussed in Sec. C.1 and *split-only* densification (as in the main ablation and our framework). Our scene extent override ablation does not change any components of the densification. Changing the extent only shifts the scale at which a Gaussian is split rather than cloned. Meanwhile, the split-only alternative modifies the densification by disabling cloning. However, it does not require determining an alternative scene extent value. Both split-only and the reduced scene extent on their own improve the StopThePop baseline by a small amount. The overall impact is minor, however, confirming that neither adjustment interferes with the default StopThePop. Adding our contributions on top of StopThePop enables densification in the occlusion-heavy Zip-NeRF scenes, where it was previously suppressed. The effect is that without any handling of the clone operation, large floater clusters appear and cause quality metrics to deteriorate significantly. This is also shown qualitatively in the first row of Fig. S1. Adding our contributions on top of StopThePop with either of the two clone corrections (scene extent override or split-only) pre-



Fig. S1: The clone failure mode on a nyc test view (30k schedule; one of the three ablation runs). Top row: StopThePop (25.5 dB) reconstructs the view, then + *contributions* collapses it into scene-covering floater clusters (12.3 dB). Bottom row: + *split-only* behaves like StopThePop (25.7 dB), then + *contributions* + *split-only* resolves the view best (28.5 dB). Insets (bottom): our contributions recover the individual keyboard keys and near-legible on-screen text that split-only densification alone leaves blurred.

vents the catastrophic failure caused by the clone operation. In both cases, the no-longer-suppressed densification allocates more Gaussians to starved regions, which improves all quality metrics over the StopThePop baseline (0.6 to 0.7 dB PSNR and roughly 9% LPIPS). Figure S1 gives a qualitative comparison for the split-only variant in the second row, with zoomed crops in the third row. Our contributions clearly sharpen fine detail that split-only densification alone leaves blurred.

D Occlusion Rate of the Datasets

Our contributions build on occlusion-aware visibility, so their impact should depend on the rate of occlusions in a scene. We determine the rate of occlusions

in the evaluated datasets in Sec. D.1 and correlate the impact of our proposed contributions with this occlusion rate in Sec. D.2.

D.1 Definition and Measurement of Occlusions

A Gaussian is occluded if it passes the frustum-based visibility filter while failing our occlusion-aware visibility filter. Let $f_{i,v} = 1$ if Gaussian i passes the frustum filter in view v and 0 otherwise, and let $r_{i,v} = 1$ if i also passes the occlusion-aware filter in v, that is, if it is rendered to at least one pixel of the output (otherwise 0). We define the per-scene occlusion rate as

$$o = 1 - \frac{\sum_v \sum_i r_{i,v}}{\sum_v \sum_i f_{i,v}}. \tag{S6}$$

It is the share of frustum-visible Gaussian observations that do not actually contribute, computed across all training views of a scene. This is precisely the set of observations that our occlusion-aware filter excludes from the densification statistics.

This approach of measuring occlusions requires a Gaussian model to render from. We considered two different types of models: initializations, which are method-independent, and fully optimized models (30k iterations), which have the most complete geometry but depend on the method used to produce them. We tried initializations with both the original initial opacity of 0.1 and a raised opacity of 0.75. Both initializations are unreliable and fail in opposite directions: 0.1 returns an occlusion rate below 0.02 for 13 of the 17 scenes. At 0.75, four scenes (including Mip-NeRF360 scenes) exceed an occlusion rate of 0.90. While a low opacity obviously reduces occlusions, we believe that *e.g.* floaters in the sparse point cloud can exaggerate occlusions in the case of high opacity initialization. When evaluating occlusions on fully optimized models, there might be a bias depending on the method that was used to optimize the model. To avoid this bias, we compared occlusion rates between models produced by five different methods: StopThePop (Reset), StopThePop (Decay), PixelGS, Grubert *et al.*, and ours. We found that the occlusion rankings correlate strongly (Spearman 0.92–0.99), which indicates that the occlusion rate is a property of the scenes rather than the methods used to optimize the model. Therefore, we choose to report occlusion rates measured on the fully optimized models, averaged over the five methods.

Table S6 reports the occlusion rates per dataset. The four Zip-NeRF apartments are the most occluded at 0.63, followed by Tanks & Temples at 0.48, while Mip-NeRF360 and Deep Blending are far lower at 0.19 and 0.15. This ordering follows the geometry: the Zip-NeRF scenes are multi-room interiors in which most surfaces are hidden behind walls from any given camera, whereas the Mip-NeRF360 and Deep Blending scenes are largely centered on a single object or a single open space. Tanks & Temples seems like an outlier: despite containing small scenes with a camera orbiting an object, it has a large occlusion rate. The reason is that the objects in this dataset (a truck and a train) are very large such that they occlude significant amounts of the scene from any given camera.

Table S6: Occlusion rate (Eq. (S6)) per dataset, averaged over five trained models and all training views. The range gives the minimum and maximum over the dataset's scenes. Training views are averages.

Dataset	Occlusion	Scene range	Train views
Zip-NeRF [2]	0.630	0.568–0.700	1336
Tanks & Temples [16]	0.482	0.388–0.577	241
Mip-NeRF360 [1]	0.194	0.090–0.342	189
Deep Blending [11]	0.145	0.083–0.208	214

D.2 Impact of Contributions Correlates with Occlusion Rate

Table S7 pairs each scene's occlusion rate with the LPIPS improvement our two contributions deliver over the occlusion-blind baseline. Both are measured inside the integrated framework with every other component held fixed, so the comparison isolates the contributions themselves. Across all 17 scenes, the two quantities correlate at Spearman +0.83 (Pearson +0.80).
However, we note that the Zip-NeRF scenes were only optimized for 30k iterations, which we find to be insufficient (more details on the training schedule in Sec. G). The quality metrics might, therefore, also be affected by insufficient reconstruction on those four scenes. Considering only the other 13 scenes still results in a Spearman correlation of +0.81 (Pearson +0.84). The largest relative gains are on the two scenes from Tanks & Temples, which are the most occluded besides the Zip-NeRF dataset.

E Occluded-Gaussian Pruning

In this section, we detail our contribution prune in Sec. E.1. We then evaluate how many fully occluded Gaussians remain in the final reconstructions of different methods in Sec. E.2.

E.1 Contribution-Prune Implementation

We employ contribution pruning, as used by RadSplat [19] and ReLaGS [26]. However, our pruning runs alongside training, similarly to OC2DGS [21] and EAGLES [8]. We improve the scheduling to be automatically synchronized with the training-view queue: a pruning decision is made only after every view has been seen once since the last decision. This prevents Gaussians in rarely observed regions from getting pruned due to a lack of opportunity. We further integrate the pruning with the existing opacity prune, which avoids additional pruning calls due to a misalignment of the training queue and densification intervals. This is done by recording a pruning mask when the pruning decision can be made, and postponing the actual pruning to the next densification event. After densification finishes, contribution pruning continues to remove redundant Gaussians on its own. Lastly, a post-training step iterates once through all training

Table S7: Per-scene occlusion rate against the LPIPS improvement of our two contributions over the occlusion-blind baseline. ΔLPIPS is the improvement of ours. Sorted by occlusion rate.

Dataset	Scene	Occlusion	LPIPS base	ΔLPIPS
Deep Blending	playroom	0.083	0.2291	−0.0037 (−1.6%)
Mip-NeRF360	room	0.089	0.2062	+0.0047 (+2.3%)
Mip-NeRF360	kitchen	0.144	0.1253	+0.0074 (+5.9%)
Mip-NeRF360	bicycle	0.146	0.1797	+0.0015 (+0.8%)
Mip-NeRF360	stump	0.178	0.1910	+0.0062 (+3.3%)
Mip-NeRF360	garden	0.179	0.1029	+0.0063 (+6.1%)
Mip-NeRF360	bonsai	0.201	0.1974	+0.0134 (+6.8%)
Deep Blending	drjohnson	0.208	0.2355	+0.0002 (+0.1%)
Mip-NeRF360	counter	0.225	0.1972	+0.0163 (+8.2%)
Mip-NeRF360	treehill	0.243	0.2844	+0.0109 (+3.8%)
Mip-NeRF360	flowers	0.342	0.2801	+0.0239 (+8.5%)
Tanks & Temples	truck	0.388	0.1252	+0.0186 (+14.9%)
Zip-NeRF	alameda	0.568	0.4014	+0.0253 (+6.3%)
Tanks & Temples	train	0.577	0.1927	+0.0250 (+13.0%)
Zip-NeRF	london	0.581	0.3650	+0.0271 (+7.4%)
Zip-NeRF	nyc	0.673	0.3188	+0.0186 (+5.8%)
Zip-NeRF	berlin	0.700	0.3237	+0.0177 (+5.5%)

views for a final contribution prune, ensuring no low-contribution Gaussians remain from the final training iterations. Overall, our contribution prune contains only a single hyperparameter: the contribution threshold. We conservatively set it equal to the opacity prune's 0.005 threshold. *I.e.* for fully visible foreground Gaussians, it does not prune anything that the opacity prune would not already prune. However, it extends this to partially and fully occluded Gaussians, while restricting pruning to those that had a chance to be seen.

E.2 Occluded-Gaussian Prevalence

Table S8 shows the share of occluded Gaussians relative to the total Gaussian count in the final reconstructions of state-of-the-art methods and ours. Occluded Gaussians are those that are fully occluded, *i.e.* those with a maximum contribution of zero across all training views. Ours shows no occluded Gaussians in the final reconstructions, as expected given the design of our contribution prune. Meanwhile, other methods carry 10.5–15.7% occluded Gaussians on the occlusion-heavy Zip-NeRF scenes and 6.9–9.2% on the occlusion-light Mip-NeRF360 scenes. This represents a substantial amount of waste and highlights the importance of contribution-based pruning. The ablation study of the main paper (Sec. 4.7) also shows that this does not negatively affect the quality of the final model at our conservative threshold of 0.005. We additionally evaluate ours without the contribution prune to isolate its impact on our approach. We note that even without the contribution prune, ours contains the smallest share

Table S8: Fully occluded Gaussians (those that never contribute to any training view) in each method's reported model, on the Zip-NeRF (top) and Mip-NeRF360 (bottom) scenes (3-run averages). The occlusion-blind methods carry substantial dead weight, up to 23% on the occlusion-heavy scenes. Our contribution prune removes them during training, so the final models contain none.

	Method	G(M)	Occluded	Share
Zip	StopThePop (Reset)	0.951	149k	15.7%
	StopThePop (Decay)	**0.486**	**50.9k**	10.5%
	PixelGS [31]	1.980	449k	22.7%
	Grubert *et al.* [9]	1.131	156k	13.8%
	Ours (no contribution prune)	1.245	89.1k	**7.2%**
	Ours (Reset)	1.991	0	0%
	Ours (Decay)	1.133	0	0%
Mip	StopThePop (Reset)	3.289	296k	9.0%
	StopThePop (Decay)	**1.711**	146k	8.5%
	PixelGS	5.328	492k	9.2%
	Grubert *et al.*	3.263	225k	6.9%
	Ours (no contribution prune)	2.023	**33.3k**	**1.6%**
	Ours (Reset)	3.913	0	0%
	Ours (Decay)	1.954	0	0%

of fully occluded Gaussians in the final reconstructions (7.2% on Zip-NeRF and 1.6% on Mip-NeRF360). This indicates that our proposed average-contribution criterion makes better and more efficient densification decisions with less waste. We emphasize that this is not merely an artifact of our method being more compact than the others: on Zip-NeRF, the method by Grubert *et al.* produces final reconstructions with 9% fewer total Gaussians but 75% more occluded Gaussians. An example of the same can be seen on the Mip-NeRF360 results: StopThePop (Decay) contains 15% fewer total Gaussians while carrying more than four times as many occluded Gaussians as ours without the contribution prune.

F Average Contribution Criterion

In this section, we present a deeper analysis of our proposed average-contribution criterion: we calibrate its densification threshold (Sec. F.1), examine the denominator floor (Sec. F.2), report the full Mip-NeRF360 results (Sec. F.3), analyze where the contributions redistribute Gaussians (Sec. F.4), examine robustness to the threshold (Sec. F.5), isolate a hybrid of PixelGS and our criterion (Sec. F.6), and test alternative normalizers (Sec. F.7).

Table S9: Threshold calibration of our criterion on the Mip-NeRF360 calibration subset (bicycle, bonsai, counter, garden), single runs, using our full integrated configuration: the occlusion-aware filter, the average-contribution criterion, and the adopted densification components. Our operating point 0.002 (bold) attains the best PSNR, SSIM, and GMSD of the sweep; over- and under-densification degrade quality on either side. Counts are in millions of Gaussians.

Threshold	PSNR↑	SSIM↑	LPIPS↓	GMSD↓	Count
0.001	27.98	0.864	0.162	0.077	4.90
0.0015	28.25	0.874	**0.157**	**0.073**	2.75
0.002	**28.31**	**0.875**	0.160	**0.073**	1.79
0.0025	28.24	0.874	0.166	0.074	1.30
0.004	28.16	0.865	0.191	0.076	0.66

F.1 Threshold Calibration

We calibrate our densification threshold once, with a single-run sweep of our full configuration on the calibration subset (Tab. S9). The subset consists of four representative Mip-NeRF360 scenes: bicycle, bonsai, counter, and garden. The two outdoor scenes feature a prominent central object, one of them with thin structures, and the two indoor scenes contain a single object and multiple objects, respectively. The threshold 0.002 sits at an interior optimum, attaining the best PSNR, SSIM, and GMSD of the sweep at 1.79M Gaussians. Quality degrades on both sides, as over-densification (0.001, 4.90M) lowers PSNR and under-densification (0.004, 0.66M) raises LPIPS. We adopt this threshold and leave it unchanged on every other scene and dataset.

F.2 The Denominator Floor

Our criterion normalizes the accumulated gradient by $1 + \sum_v \bar{w}_{i,v}$, where the constant is a fixed floor (Sec. A.1). A vanishing accumulated contribution is not the failure case: a Gaussian that was never observed accumulates no gradient either, and is never densified. The failure case lies strictly in cases where a Gaussian has only few observations with low contributions, causing the accumulated contribution to fall between 0 and 1. There the accumulated contribution divides the gradient sum by less than one, inflating the score instead of normalizing it, and this inflation grows without bound as the sum approaches zero. So, densification stops responding to the signal the criterion is meant to measure. We sweep the floor over 0, 0.5, 2 and 4 on the Zip-NeRF scenes to establish two things: whether the floor is necessary, and what effect its specific value has.

Without the floor, training diverges. We evaluate the behavior of the criterion without a floor in Tab. S10. At floor 0 and our default threshold, all four scenes cross the $2^{24} \approx 16.8$M trainable limit and crash within roughly 1,800 of the 30,000 iterations. Raising the threshold delays the divergence but does not remove it: the

Table S10: Floor 0 across a threshold ladder on the Zip-NeRF scenes (30k, single runs). Diverging rows report how many scenes crossed the trainable limit, together with the peak count reached and the iteration by which training stopped. The last two rows give floor 1 for reference: our operating point, and the setting whose count is closest to the only floor-0 threshold that completes. Counts are in millions of Gaussians.

Floor	Threshold	PSNR↑	SSIM↑	LPIPS↓	GMSD↓	Count
0	0.002	diverges on 4/4 scenes; peak 17.7–21.0M by iteration ∼1,800				
0	0.004	diverges on 4/4 scenes; peak 17.4–20.1M by iteration ∼2,300				
0	0.01	diverges on 4/4 scenes; peak 17.2–17.9M by iteration ∼5,000				
0	0.02	diverges on 3/4 scenes; berlin completes at 2.00M				
0	0.03	25.15	0.803	0.369	0.123	1.63
0	0.05	24.36	0.775	0.417	0.150	0.21
0	0.25	22.94	0.746	0.455	0.175	0.04
1	0.002	25.80	0.830	0.330	0.106	1.13
1	0.0015	25.87	0.835	0.319	0.102	1.77

criterion still diverges on all four scenes at five times the default threshold, and on three of four at ten times the default. Only at fifteen times the default (0.03) does training complete on every scene. However, the resulting model is far off the quality our criterion reaches at a comparable count: at 1.63M Gaussians it loses 0.72 dB and 0.050 LPIPS against floor 1 at 1.77M. The unbounded criterion's unstable performance is therefore not a matter of calibrating the right threshold. This indicates the necessity of the denominator floor.

Above zero, the floor rescales the threshold. For the remaining floors, training is stable and the floor acts as a reparameterization of the threshold. At the default threshold (upper block of Tab. S11), the final Gaussian count increases for lower floors and decreases with higher floors. The quality moves with the count exactly as it does when the threshold itself is varied.

At a matched count, the floor value stops mattering. To separate this count effect from a genuine quality effect, we re-run each floor with a predicted threshold that should move the final count closer to the floor 1 baseline. For simplicity, we then compare these to the closest matching count from a floor 1 threshold sweep (lower block of Tab. S11). For all three tested floors, we find a floor 1 baseline within 3.8%–21.5% of the Gaussian count. Quality between these differs by only 0.01–0.08 dB PSNR and 0.001–0.007 LPIPS. All three trace the same count–quality curve as floor 1. Any floor large enough to keep the denominator from inflating the score is thus a reparameterization of the densification threshold rather than a quality parameter. We keep the floor at 1 for all experiments, since it is the minimal value that guarantees that the denominator will not fall below 1. We treat the densification threshold as the single tunable parameter of the criterion.

Table S11: The denominator floor on the Zip-NeRF scenes (30k, single runs). Upper block: at the default threshold, the floor acts as a count–quality trade off. Lower block: with each floor's threshold scaled by a predicted value, interleaved with a threshold sweep of floor 1 and ordered by count. Counts are in millions of Gaussians.

Floor	Threshold	PSNR↑	SSIM↑	LPIPS↓	GMSD↓	Count
0.5	0.002	25.83	0.836	0.314	0.100	2.39
1	0.002	25.80	0.830	0.330	0.106	1.13
2	0.002	25.58	0.818	0.353	0.117	0.52
4	0.002	25.11	0.801	0.381	0.132	0.24
1	0.004	25.44	0.812	0.363	0.121	0.40
0.5	0.004	25.71	0.824	0.342	0.111	0.78
1	0.0025	25.74	0.825	0.340	0.110	0.81
1	0.002	25.80	0.830	0.330	0.106	1.13
2	0.001	25.88	0.833	0.323	0.103	1.44
4	0.0005	25.86	0.834	0.320	0.103	1.65
1	0.0015	25.87	0.835	0.319	0.102	1.77
1	0.001	25.82	0.837	0.309	0.099	3.29

Table S12: Densification criteria on Mip-NeRF360 (same criteria as Tab. 1 of the main paper; 3-run averages) and the occlusion-blind count of the integrated framework for reference. Counts are in millions of Gaussians; time in minutes.

Method	PSNR↑	SSIM↑	LPIPS↓	GMSD↓	Count↓	Time↓
Occlusion-blind count	27.29	0.821	0.196	0.087	2.117	**31.4**
Occlusion-aware count	**27.33**	**0.822**	0.188	**0.086**	2.595	35.4
PixelGS	27.22	0.819	0.184	0.087	3.433	43.8
PixelGS + DS	27.20	0.820	**0.183**	0.087	3.157	42.2
Ours	27.30	0.820	0.186	0.087	**1.954**	35.4

F.3 Results on Mip-NeRF360

In this section, we present the result tables for the Mip-NeRF360 scenes whose numbers are discussed in the main paper. Table S12 shows the Mip-NeRF360 results corresponding to the Zip-NeRF results reported in Tab. 1 of the main paper.

Ours is the only criterion that ends below the occlusion-blind count it is built on, at 1.954M Gaussians against 2.117M, while every count-based criterion grows the model instead: 2.595M for the occlusion-aware count, and 3.157–3.433M for the two PixelGS variants. Despite being more compact, quality does not suffer for ours: against the occlusion-blind count, ours improves LPIPS by 5% at a PSNR difference of 0.01 dB. Against the best LPIPS in the table it gives up 0.003 while using 38% fewer Gaussians. SSIM and GMSD barely move at all, spanning 0.819–0.822 and 0.086–0.087 across the table, which indicates that there are no noteworthy occlusion-induced errors in these lightly occluded scenes (Sec. D).

Table S13: Mip-NeRF360 performance split by scene type (3-run averages): the full 9-scene average, the 4 indoor scenes (bonsai, counter, kitchen, room), and the 5 outdoor scenes (bicycle, flowers, garden, stump, treehill). Counts are in millions of Gaussians.

	Method	PSNR↑	SSIM↑	LPIPS↓	GMSD↓	Count↓
Full	StopThePop	27.01	0.809	0.227	0.102	1.711
	Occlusion-blind count	27.29	0.821	0.196	0.087	2.117
	Occlusion-aware count	**27.33**	**0.822**	0.188	**0.086**	2.595
	Ours	27.30	0.820	**0.186**	0.087	**1.954**
Indoor	StopThePop	30.16	0.917	0.194	0.064	0.756
	Occlusion-blind count	30.60	0.922	0.182	0.060	**0.741**
	Occlusion-aware count	30.72	0.924	0.177	0.058	0.852
	Ours	**30.87**	**0.927**	**0.171**	**0.056**	0.812
Outdoor	StopThePop	24.49	0.722	0.254	0.133	2.475
	Occlusion-blind count	**24.65**	**0.741**	0.208	**0.109**	3.218
	Occlusion-aware count	24.61	0.740	**0.197**	**0.109**	3.989
	Ours	24.44	0.734	0.198	0.112	**2.868**

We additionally present an evaluation of the difference between indoor and outdoor scenes of Mip-NeRF360 in Tab. S13, as is done by related work [12,20]. We note that ours performs generally better on the indoor scenes where most of the scene consists of well-defined geometry. On outdoor scenes where a lot of distant objects exist which are not observed well and contain a lot of foliage, ours drops a little in PSNR, SSIM, and GMSD but stays roughly on par with the occlusion-aware count in LPIPS. In terms of Gaussian count, ours is more compact than the occlusion-aware count on every split and trails only the vanilla StopThePop baseline overall, which is also the weakest in quality. We provide an additional evaluation of the decreased outdoor quality metrics in Sec. F.4, where we show that these pixel-aligned metrics are less reliable than the perceptual LPIPS on such foliage-heavy scenes.

F.4 Spatial Gaussian Allocation

We analyze *where* the occlusion-aware filter and average-contribution criterion redistribute Gaussians. Figure S10 maps the top-down Gaussian density of two representative Mip-NeRF360 scenes, comparing the occlusion-blind count, the occlusion-aware count, and our average-contribution criterion on the same integrated baseline. We choose the Mip-NeRF360 dataset as it is one that all methods are able to reconstruct well enough to isolate the effect of our contributions absent other problems such as camera coverage (Fig. S3) or exposure and lighting changes [19]. On the well-observed indoor object (kitchen) the criterion concentrates Gaussians on the central region that the dense observations can reconstruct most reliably. Its higher count (0.97M against 0.73M for the blind count) therefore places Gaussians where the observations are most informative. This is also reflected in its overall better quality metrics on the indoor

Table S14: Region-based allocation and image quality on the flowers scene (3-run averages). The two regions are the flower bed and the surrounding grass ring (radius < 1.9 and 1.9–2.9 from the centre of the flower bed in the top-down projection, respectively). For each region, Count is the number of Gaussians (in millions) positioned within it, and PSNR/SSIM/GMSD are measured on rough image crops of those regions across all test views.

	Flower bed				Grass ring			
Criterion	Count	PSNR↑	SSIM↑	GMSD↓	Count	PSNR↑	SSIM↑	GMSD↓
Blind	0.26	20.52	0.694	0.169	0.18	**19.49**	**0.279**	0.216
OA-count	0.42	**20.77**	0.712	0.163	1.80	19.42	0.265	**0.202**
Ours	0.97	20.71	**0.719**	**0.159**	0.62	18.95	0.243	0.205

scenes. On the foliage-heavy outdoor scene (flowers) the occlusion-aware count over-densifies the flat lawn around the bed, which our criterion keeps lean while allocating more Gaussians to the central flower bed. This spatial view is the counterpart of Tab. S13: on the indoor scene, ours allocates more Gaussians to the central object than the occlusion-blind count, which increases overall quality. Meanwhile, on the outdoor scene, ours needs roughly a quarter fewer Gaussians than the occlusion-aware count, which heavily over-densifies the grass around the flower bed.

Table S14 makes the disproportionate allocation explicit. In the flower bed our criterion places the most Gaussians of the three (2.3× the occlusion-aware count and 3.7× the occlusion-blind count) and reconstructs it best, with the highest SSIM and lowest GMSD. On the grass ring around the flower bed, the occlusion-aware count instead allocates ten times as many Gaussians as the occlusion-blind baseline. While it improves its GMSD slightly, both PSNR and SSIM deteriorate. The occlusion-blind count achieves the best PSNR for the grass ring although it is noticeably more blurry than the occlusion-aware count and ours (see the sample grass crop in Fig. S10). This shows that PSNR and SSIM reward the smoothing that blur produces, so the blurry occlusion-blind lawn scores highest out of the three. Meanwhile, GMSD penalizes blur and ranks it worst. However, due to GMSD still being pixel-aligned, it marginally favors the occlusion-aware count's denser grass over ours.

F.5 Robustness

We assess how sensitive our results are to two protocol choices: the densification threshold each criterion is given, and the resolution at which we evaluate.

Densification threshold sweep. We sweep the densification threshold from values that starve the model to values that force heavy over-densification, for every criterion on a subset of the Mip-NeRF360 [1] dataset (garden, bicycle, bonsai and counter). We plot image quality against final count in Fig. S2, sweeping each criterion over $\frac{1}{2}$, 1, 2 and 4 times its own default threshold, so the threshold

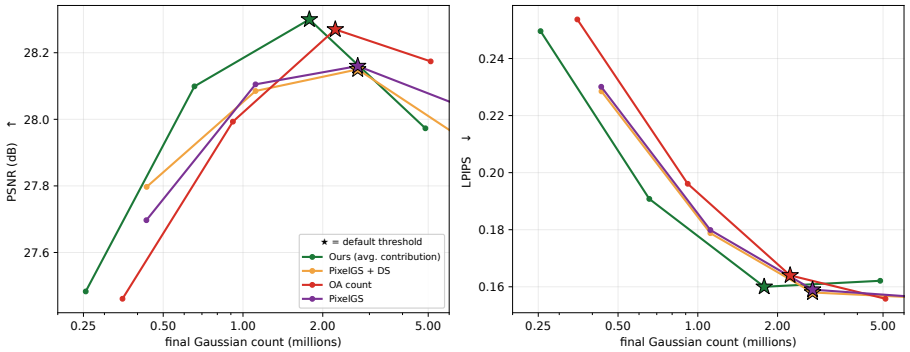


Fig. S2: Robustness to the densification threshold on the Mip-NeRF360 subset: image quality versus final Gaussian count. Each data point represents a different threshold decreasing from left to right, as the count increases. All points are three-run averages.

decreases from left to right as the count grows. Across the sweep every criterion follows the same arc: image quality rises with the Gaussian count up to a knee and then deteriorates as over-densification begins. On this subset, our average-contribution criterion reaches this knee at the *lowest* count of any criterion: its default operating point already attains the best quality of the whole sweep at only 1.78M Gaussians, so it offers the best quality per Gaussian. Ours even sits at an interior optimum: halving the threshold grows the count 2.7× while degrading both PSNR and LPIPS, whereas the count-based criteria keep improving LPIPS only by adding Gaussians while their PSNR declines. At half the default threshold they already reach roughly three times our operating count (5.1–6.3M), the raw pixel-count criterion inflating the most while distance scaling barely changes it on this dataset. At a quarter of the default threshold every criterion over-densifies past the $2^{24} \approx 16.8$M trainable limit and diverges on both outdoor scenes of the subset, which is why that setting is absent from the plot. So the meaningful comparison is at each criterion's operating point, where ours is simultaneously the most compact and the highest quality.

Native resolution. Following the common protocol, we evaluate the Zip-NeRF scenes at half resolution (Sec. 4.1 of the main paper). Since downsampling removes exactly the fine detail our contributions are meant to recover, we repeat the 30k comparison against vanilla StopThePop [20] at the native, full resolution of the dataset (Tab. S15). The resolution does not change the outcome: ours improves every metric on all four scenes. The PSNR margin is in fact slightly larger than at half resolution (+0.91 against +0.86 dB). The perceptual margins shrink but remain substantial, with LPIPS improving by 7.5% instead of 12.2% and GMSD by 16% instead of 20%. Ours achieves this with 2.1× the vanilla count, against 2.3× at half resolution. The reported gains are therefore a property of the densification, not of the evaluation resolution.

Table S15: Zip-NeRF trained with full-resolution images against the half-resolution protocol used throughout the paper (30k, single runs). Shading compares within each resolution block. Counts are in millions of Gaussians.

	Method	PSNR↑	SSIM↑	LPIPS↓	GMSD↓	Count↓
Full	StopThePop	24.67	0.813	0.372	0.152	**0.396**
	Ours	**25.58**	**0.833**	**0.344**	**0.128**	0.832
Half	StopThePop	24.95	0.802	0.376	0.133	**0.486**
	Ours	**25.81**	**0.830**	**0.330**	**0.106**	1.133

F.6 The PixelGS-ours Hybrid

Our criterion and the PixelGS [31] criterion differ in two aspects. The first is the per-view weight: our average contribution versus PixelGS's visible-pixel count. The second is how that weight enters the update: PixelGS forms a weighted mean of the per-view gradients, whereas we sum the gradients and normalize by the accumulated average contributions. To isolate the aggregation rule, we evaluate the PixelGS-ours hybrid: PixelGS's weighted-mean structure fed our contribution signal, using each view's total contribution $W_{i,v} = \sum_p w_{i,p}$ as the weight instead of the visible-pixel count, *i.e.* $\gamma_i^W = \left(\sum_v W_{i,v} \, \|g_{i,v}\|\right) / \sum_v W_{i,v}$. As this hybrid has no published default, we calibrate its threshold on the Mip-NeRF360 subset of the threshold sweep (Sec. F.5), selecting 0.0003. Table S16 repeats the Zip-NeRF criterion comparison with this arm added (three runs). The transmittance weighting does not repair the weighted-mean aggregation: at its operating point the PixelGS-ours hybrid lands almost exactly on the raw PixelGS point (6.4M versus 6.6M Gaussians at near-identical quality and training time), over-densifying heavily. The same contribution signal, entering instead through our normalizer, stays far more compact at 1.1M. Under the extended 100k schedule of the main paper, the PixelGS-ours hybrid diverges on three of the four scenes and reaches 7.4M Gaussians on the fourth (single runs). This is the same long-schedule instability as the raw pixel-count criterion.

F.7 Alternative Normalizers

Our criterion is one of a family of possible contribution-based normalizers, so we test some plausible alternatives explicitly. The *per-view* variant divides both sums by the number of rendered views N_i:

$$\gamma_i^{\text{per-view}} = \frac{\sum_v \|g_{i,v}\|/N_i}{1 + \sum_v \bar{w}_{i,v}/N_i}. \tag{S7}$$

The *sum-allocation* variant replaces the per-view mean by the per-view total $W_{i,v} = \sum_p w_{i,p}$:

$$\gamma_i^{\text{sum-alloc}} = \frac{\sum_v \|g_{i,v}\|}{1 + \sum_v W_{i,v}}. \tag{S8}$$

Table S16: The PixelGS-ours hybrid on the occlusion-heavy Zip-NeRF scenes, added to the criterion comparison of the main paper (calibrated threshold 0.0003, three runs), preceded by the occlusion-blind count of the integrated baseline; the extended-schedule row reports single runs. Counts are in millions of Gaussians; time in minutes.

Method	PSNR↑	SSIM↑	LPIPS↓	GMSD↓	Count↓	Time↓
Occlusion-blind count	25.52	0.818	0.352	0.117	**0.557**	**22.3**
Occlusion-aware count	25.75	**0.834**	0.319	**0.103**	2.567	36.6
PixelGS	25.11	0.829	**0.312**	0.105	6.607	67.0
PixelGS + DS	25.59	0.824	0.341	0.112	1.443	28.8
PixelGS-ours hybrid	25.18	0.828	0.313	0.104	6.366	65.2
Ours	**25.81**	0.830	0.330	0.106	1.133	29.9
PixelGS-ours hybrid (100k)	diverges on 3 of 4 scenes					

Table S17: Alternative normalizers on the Zip-NeRF scenes (30k, single runs), swept to bring each into our count range; our operating point is the 3-run average of the main paper. Counts are in millions of Gaussians.

Normalizer	Threshold	PSNR↑	LPIPS↓	GMSD↓	Count
per-view	0.002	23.80	0.431	0.162	0.06
	0.0005	25.33	0.366	0.125	0.45
	0.00025	25.74	0.329	0.107	1.60
sum-allocation	0.0003	22.86	0.455	0.176	0.12
	0.0002	22.93	0.449	0.173	0.89
	0.0001	diverges on 3/4 scenes; berlin completes at 1.79M			
Ours	0.002	25.81	0.330	0.106	1.13

Both place the denominator on a different scale than ours, so we run a threshold sweep for each such that it brings the final count into our range (Tab. S17). Neither improves on our normalizer. The per-view variant is similar in quality to ours but requires 1.4× as many Gaussians. Sum-allocation fails outright. A slight change in the threshold to push its count closer to ours causes it to diverge on three of the four scenes: each crossing the $2^{24} \approx 16.8$M trainable limit, as the unfloored criterion shows at this schedule (Sec. F.2). The pixel-count criterion from PixelGS reaches the same limit under the longer schedules of Sec. G.3. The single scene that completes overshoots our count by 69% while remaining 2.77 dB behind. This is precisely the failure anticipated in Sec. A.1: the per-view total grows with the on-screen footprint, reintroducing the pixel-count dependence that the per-view mean removes.

G Scalability under a Matched Training Budget

In this section, we analyze the effect of extended training schedules on the reconstruction performance. Specifically, the larger Zip-NeRF scenes contain an order



Fig. S3: Training-camera distribution on the alameda scene (top-down). Dots mark the COLMAP camera positions, colored by the local density of training cameras; the sparse reconstruction is shown underneath for orientation. Interior hotspots are observed by an order of magnitude more cameras than the garden loop or the peripheral rooms.

of magnitude more images than the smaller Mip-NeRF360 scenes. This could reasonably result in an imbalance in the results between the small scenes and large scenes due to missing opportunity. Extending the training schedule isolates the opportunity and shows each method's performance ceiling. We first map the uneven camera coverage of the large scenes (Sec. G.1), then detail the schedule scaling and report the 300k results (Sec. G.2), trace the Gaussian count over iterations (Sec. G.3), examine the distance-scaling guard rail (Sec. G.4), and close with per-view statistics (Sec. G.5).

G.1 Camera Distribution

Figure S3 maps the training-camera distribution of the alameda scene, extracted from its COLMAP reconstruction. Coverage is starkly imbalanced: some interior hotspots are covered by an order of magnitude more cameras than the garden or peripheral rooms. This is consistent with the per-view analysis (Sec. G.5), whose weakest views lie in exactly such under-observed regions, and supports camera distribution as a remaining quality factor beyond occlusion.

Table S18: Extended 300k schedule on the Zip-NeRF scenes (single runs), continuing the main paper's extended-training comparison. Counts are in millions of Gaussians; time in minutes.

Method	PSNR↑	SSIM↑	LPIPS↓	GMSD↓	Count↓	Time↓
PixelGS + DS	26.46	0.848	0.285	0.090	4.472	553.5
Ours	**26.59**	**0.852**	**0.278**	**0.086**	**2.754**	**395.7**

G.2 Extended Schedules and 300k Results

For the matched 210k schedule of the main paper and the 100k and 300k schedules sampled around it, we lengthen the training schedule and adjust its hyperparameters accordingly, and we report single runs owing to their cost. Relative to the 30k base, we scale the schedule by a common factor $s = 10/3$, 7, and 10 for the 100k, 210k, and 300k schedules. The iteration count becomes $s \times 30{,}000$. The positional learning-rate schedule, the densification window (iterations 500 to 15,000 at the 30k base), the opacity-reset interval, and the opacity-decay interval all scale by the same factor of s. The densification interval is instead held fixed at 100 iterations. This ensures that the densification opportunity per image also scales with s. Table S18 extends the main paper's extended-training comparison to a 300k schedule: ours stays bounded (only 10% more Gaussians than at 210k) with LPIPS still improving marginally, while the guarded PixelGS [31] also remains bounded but at 62% more Gaussians and 40% more training time than ours.

G.3 Count vs. Iteration

We evaluate the trajectory of the Gaussian counts over the training duration. Figure S4 traces the count per iteration for all four criteria on the four Zip-NeRF scenes, at the 210k schedule and, for the two criteria that complete it, also at 300k. The occlusion-aware filter and all other components of the integrated framework are held fixed, isolating the criteria's influence alone.

Ours is the only criterion that stays bounded without a guard rail. It grows from 1.3–1.6M Gaussians at iteration 30k to a peak of 2.6–3.3M at the end of the densification window. The count then declines monotonically to 2.2–2.8M, with it flattening well before the training ends. This decline is the contribution prune, which keeps removing occluded primitives once densification no longer replaces them (Sec. E.1).

The two count-based criteria diverge instead. The PixelGS criterion without the distance scaling crosses the $2^{24} \approx 16.8$M trainable limit (Sec. F.2) on all four scenes and completes none of them. The occlusion-aware count peaks later than the PixelGS criterion but does not avoid the divergence: it crashes on two of the four scenes. Notably, on the scenes where it does not crash it peaks at counts that are 1.8–2.5× the final count. Since all criteria are using the same framework in these experiments, the occlusion-aware count also benefits from the contribution

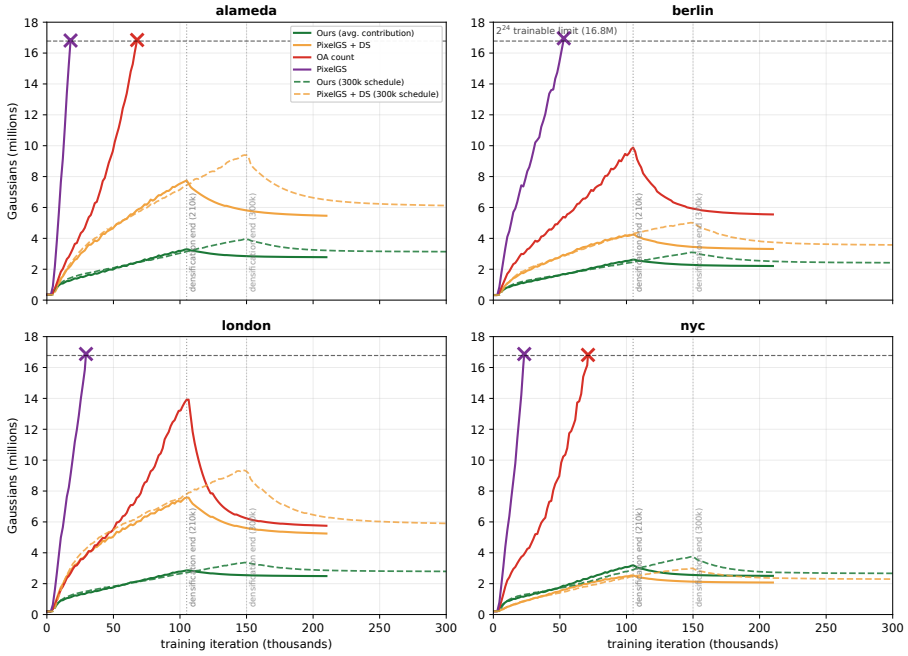


Fig. S4: Gaussian count versus iteration on the four Zip-NeRF scenes (single run per scene, 210k solid, 300k dashed, dotted verticals mark each schedule's densification end, crosses mark crashes at the trainable limit). Ours (green) and PixelGS+DS (orange) stay bounded. The PixelGS criterion (purple) itself diverges on every scene, while the occlusion-aware count (red) diverges on two scenes.

prune. The large amount of pruning indicates that a significant amount of Gaussians that are created are redundant and not beneficial to the overall geometry or appearance of the reconstruction. Conversely, this also indicates that ours is more efficient in choosing Gaussians to densify as the amount of pruning post densification is comparatively small. Despite heavy pruning, the occlusion-aware count still ends at 2.3–2.5× our count.

The PixelGS [31] criterion with distance scaling (PixelGS+DS) is bounded like ours, though by an additional guard rail than by the criterion itself. It still ends at 1.6× our count on average and pays for it by suppressing the near-field densification whose absence shows up as blur (Sec. G.4).

G.4 The Distance-Scaling Guard Rail

The distance-scaling guard rail of PixelGS [31] suppresses the densification gradient of near Gaussians quadratically with depth, scaling it by $\min\big((z/(\beta E))^2, 1\big)$ with camera-space depth z, constant β, and scene extent E; the wide Zip-NeRF camera orbits inflate E, so close-up detail is systematically under-densified and

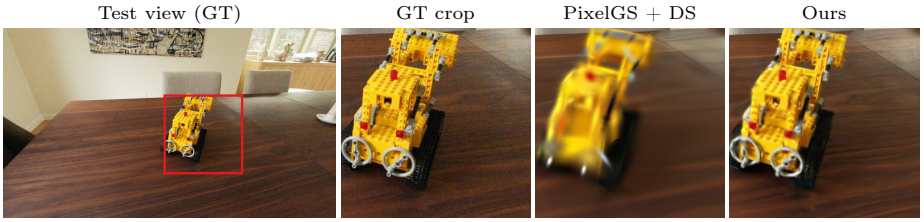


Fig. S5: Distance scaling (DS) suppresses near-object densification (alameda, 300k schedule). On this close-up test view, PixelGS with DS (6.1M Gaussians) leaves the near object a featureless blur, whereas our average-contribution criterion (3.1M Gaussians) reconstructs the detail well.

left blurry (Fig. S5)—the same weak scene-extent estimate we argue against for cloning and pruning. Our criterion requires no such term: unlike a pixel count, its per-view mean contribution is independent of the on-screen footprint, so near views are never up-weighted in the first place, and near and far balance without a distance term. The pixel-count criterion thus over-densifies without the guard (Fig. S4) and suppresses fine detail with it, whereas ours is stable and detail-preserving with no guard at all (per-view breakdown in Sec. G.5).

G.5 Per-View Analysis

Averaged over full test sets the distance-scaling asymmetry appears modest (Tab. 2 of the main paper) because it is spatially concentrated: on alameda at 210k, ours improves LPIPS over PixelGS + DS on 133 of 217 test views by more than 0.005, with the largest margins on the near-object close-ups (up to 0.10 LPIPS and 2.4 dB better on the Lego toy). The few views favoring PixelGS + DS by that margin (27 of 217, not surpassing 0.05 LPIPS) are severely under-observed corners, where its near suppression yields an inoffensive smooth blur, while the densification ours enables appears as floater speckle (Fig. S6). We note that the ground truth images show pixel-level color artifacts, which explain the colorful speckles. Ours attempts to reconstruct this although it might be impossible to reconstruct this high-frequency noise that also changes between different viewpoints. Here, PixelGS is rewarded for its blurred reconstruction. Our criterion's lowest-quality views are intrinsically difficult views on which every method degrades together: on Mip-NeRF360 only a single test view falls below 18 dB and none exceed 0.45 LPIPS. At the default schedules all three criteria place nearly the same number of views below 18 dB (one on Mip-NeRF360; 17–18 on Zip-NeRF [2] at 30k). At 210k the occlusion-aware count instead collapses to a cluster reaching 12.5 dB on berlin and london, where ours leaves only a single view below 18 dB (minimum 16.0 dB). The 30k per-view statistics are averaged per view over the three repetitions, while the extended-schedule statistics come from single training runs.

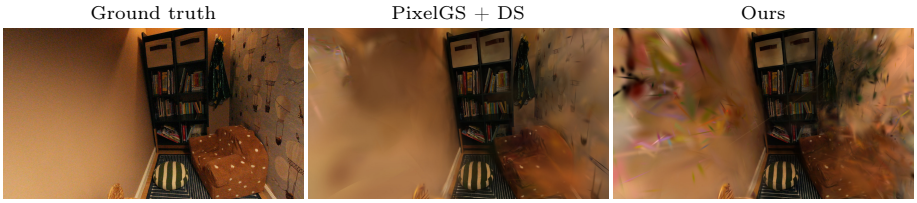


Fig. S6: An under-observed corner view (alameda test view 00070, 300k schedule). The region is not intrinsically difficult but instead under-observed. The distance-scaled PixelGS renders an inoffensive smooth blur. Ours densifies to reconstruct the pixel-level color artifacts present in the ground truth, which it renders as colorful speckle.

Per-view behavior. Our criterion also avoids method-specific worst-case failures, whereas the count-based criteria's over-densification produces isolated severe failures (*e.g.* 12–15 dB view clusters on london at 210k where ours holds 27–33 dB).
At the default schedules the three criteria's per-view PSNR distributions are statistically indistinguishable (Fig. S7). The divergence is driven by schedule length: as training lengthens the unguarded occlusion-aware count develops a growing low-quality tail (29 sub-18 dB views at 100k against 16 views for ours), whereas ours stays as bounded as the distance-scaled PixelGS [31] without needing that guard and with fewer Gaussians.

H Additional Qualitative Results

Figure S8 shows the three teaser locations of the main paper as full test frames.

We show full-frame renders and error maps against vanilla PixelGS [31] (StopThePop base) on three Zip-NeRF scenes in Fig. S9. We visualize gradient-magnitude similarity ($1 - \mathrm{GMS}$) rather than absolute error, because absolute error barely separates the two methods: the main failure is blur caused by insufficient reconstruction, which is favored by per-pixel metrics. Gradient-magnitude similarity instead responds to lost structure, which is where the two methods differ. For each scene we show the test views on which our LPIPS advantage over PixelGS is largest and smallest, respectively. Where the advantage is largest, PixelGS visibly loses fine structure such as tiles, foliage, and clutter. Its error map is correspondingly brighter: its GMSD is 0.127, 0.178, and 0.159 against our 0.088, 0.129, and 0.103. Where the advantage is smallest, the two error maps are nearly indistinguishable. On nyc our LPIPS is in fact better on every test view. Our weakest views relative to PixelGS are therefore not views where it recovers more detail, but views where the two reconstructions are structurally equivalent.

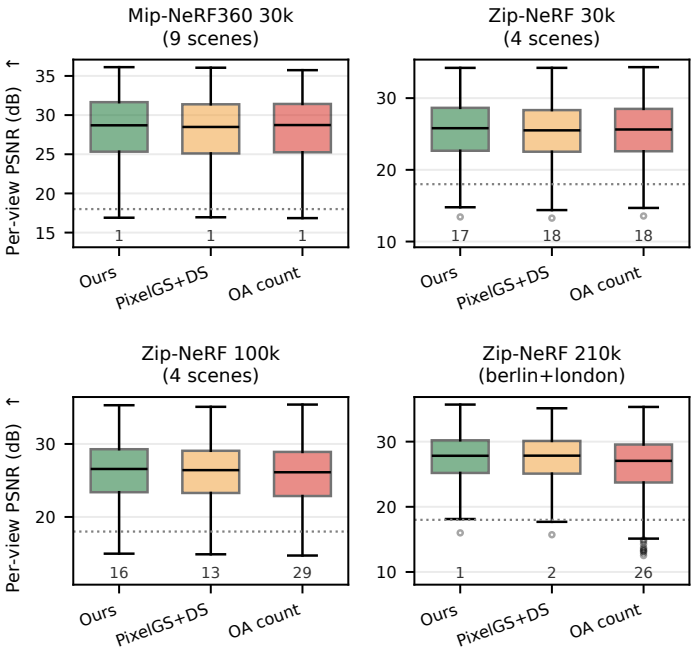


Fig. S7: Per-view PSNR distribution per criterion across four training schedules. Numbers below each box count the views under 18 dB. The 30k results are three-run averages; the extended schedules are single runs. The 210k panel restricts all criteria to berlin and london, the two scenes on which all criteria complete.

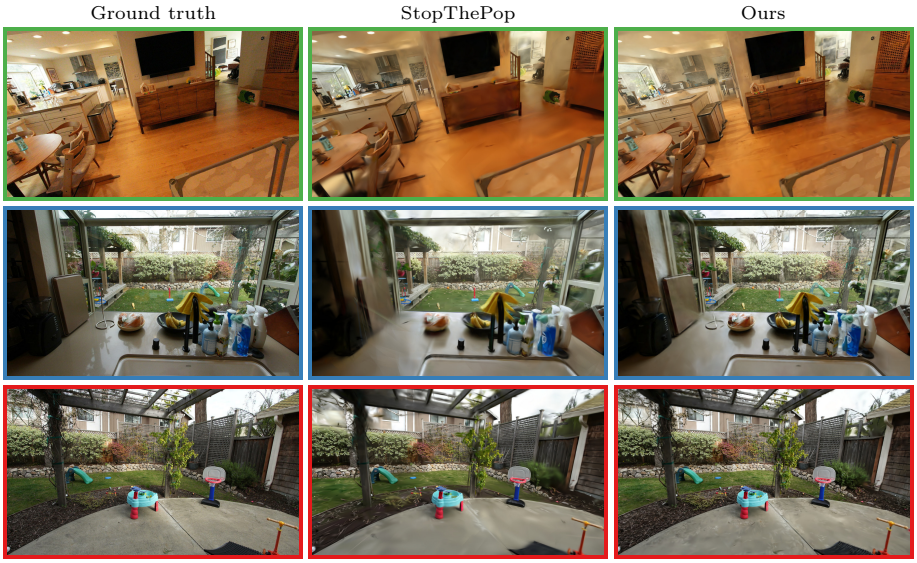


Fig. S8: The three teaser locations of the main paper (Fig. 1) as full test frames (ground truth, StopThePop, ours). Frame colors match the location boxes of the teaser.

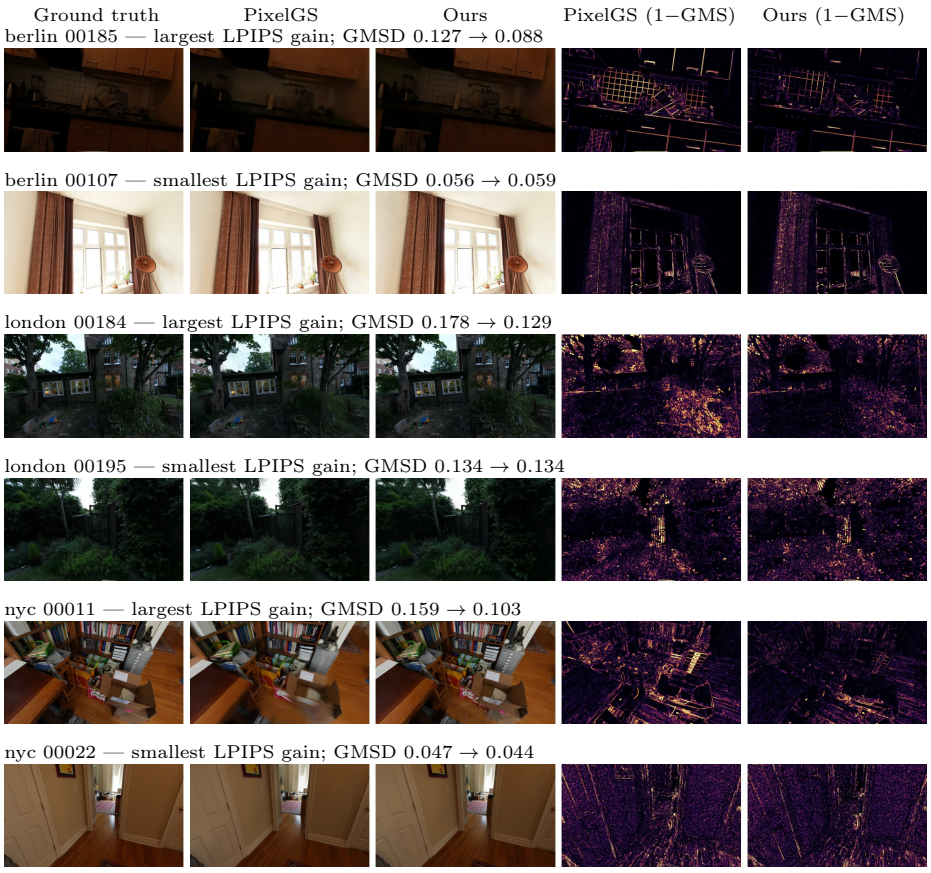


Fig. S9: Full-frame renders and error maps against vanilla PixelGS [31] on three Zip-NeRF scenes (30k, single runs). Columns: ground truth, PixelGS, ours, and the gradient-magnitude-similarity error map (1−GMS, brighter is worse) of each. For every scene, the first row is the test view on which our LPIPS advantage over PixelGS is largest and the second the view on which it is smallest. Row labels give the per-view GMSD as PixelGS → ours.

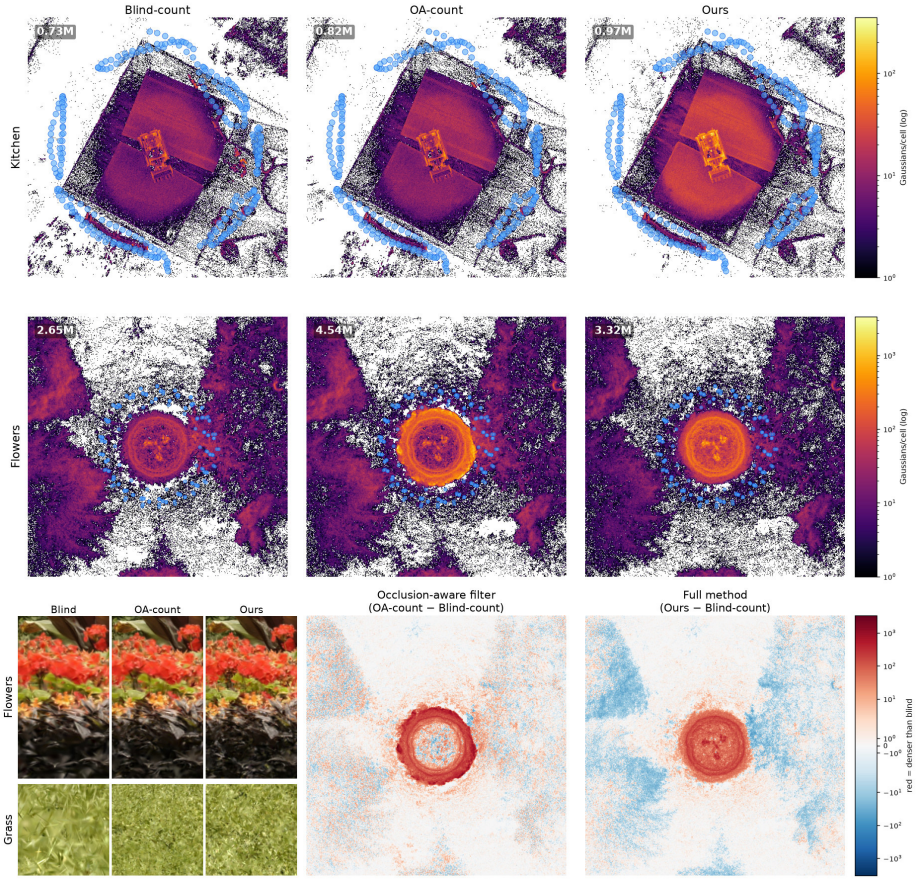


Fig. S10: Top-down Gaussian density of two Mip-NeRF360 scenes (log Gaussians per ground cell; the color scale is shared across the three criteria within each scene), comparing the occlusion-blind count (Blind), the occlusion-aware count (OA-count), and our average-contribution criterion (Ours) on the same integrated baseline. Camera positions are blue dots, and each map is annotated with its total Gaussian count. **Kitchen (top):** Ours concentrates Gaussians on the well-observed central object. **Flowers (middle):** OA-count over-densifies the flat lawn around the flower bed (its bright disk balloons outward), whereas Ours keeps the bed dense and the lawn lean. **Bottom:** render crops of a flowers test view, the flower bed (top) and the surrounding grass (bottom), each as Blind | OA-count | Ours, followed by two difference maps relative to Blind (OA-count − Blind and Ours − Blind), red denser. The former isolates the occlusion-aware filter, which adds Gaussians across the bed and the lawn; the latter shows the full effect of Ours, which concentrates them on the observed bed. Blind renders visibly blurrier grass, while OA-count and Ours are similarly sharp (Tab. S14).