A. Appendix

A.1. AccuTile Proof of Correctness Sketch

We outline the correctness of our AccuTile algorithm by examining the different cases that arise when identifying the minimum and maximum points of an ellipse within a given tile row. Due to the symmetry of Equation 14, exchanging the variables x and y along with the coefficients a and c yields equivalent statements for tile columns. Thus, we focus our discussion on tile rows.

Case 1: The ellipse does not intersect the tile row boundary. The entire ellipse, including the bounding box extrema $x_{\min}$ and $x_{\max}$ computed by SnugBox, lies within the row. AccuTile correctly selects these points as the furthest ellipse extents. Figure 4 illustrates an example of this.

Case 2: The ellipse intersects one of the tile row bounding lines but not the other.
This implies that either $y_{\min}$ or $y_{\max}$ lies within the row, but not both. There are several possible subcases:

- **Case 2.1:** If both $x_{\min}$ and $x_{\max}$ are in the tile row, then they are correctly assigned as the furthest extent of the ellipse by AccuTile. Figure 5 illustrates an example of this case.
- **Case 2.2:** If $x_{\min}$ and $x_{\max}$ are not in the row but $y_{\max}$ is, then the ellipse decreases monotonically from $y_{\max}$ to the row boundary on both sides of $y_{\max}$, making the row boundary intersections the furthest extent of the ellipse and are the points selected by AccuTile as the furthest row extent. This follows from the absence of the critical points $x_{\min}$ and $x_{\max}$. A symmetric argument applies when $y_{\min}$ is in the row instead. The top tile row of Figures 5 and 6 illustrate examples of this case.
- **Case 2.3:** If $x_{\min}$ and $y_{\min}$ are in the row but $x_{\max}$ is not, then $x_{\min}$ is assigned as the minimum extent of the ellipse. The ellipse increases monotonically from $y_{\min}$ to the boundary to the right of $y_{\min}$ and from $x_{\min}$ to the boundary to the right of $x_{\min}$. Under a corrollary of the definition of an ellipse, the sides of the ellipse do not intersect. Thus, the ellipse point on the curve that extends to the right of $y_{\min}$ and intersects the tile row boundary must be the maximum ellipse extent, and AccuTile correctly selects it as such. A similar argument applies in the following cases: (1) $x_{\max}$ and $y_{\min}$ are in the tile row but $x_{\min}$ is not, (2) $x_{\min}$ and $y_{\max}$ are in the tile row but $x_{\max}$ is not, and (3) $x_{\max}$ and $y_{\max}$ are in the tile row but $x_{\min}$ is not. The bottom tile row of Figure 6 illustrates an example of this case.

Case 3: The ellipse intersects both the top and bottom row boundary.
If $x_{\min}$ or $x_{\max}$ is in the tile row, then AccuTile correctly assigns it as the minimum or maximum extent of the ellipse, respectively. The right side of the ellipse in the middle tile row in Figure 6 illustrates an example of this case. Otherwise, the ellipse monotonically increases from the bottom row boundary to the top row boundary, or vice-versa, due to the absence of critical points. Selecting the minimum or maximum boundary point, as done by AccuTile, yields the correct result. The left side of the ellipse in the middle tile row in Figure 6 illustrates an example of this case.

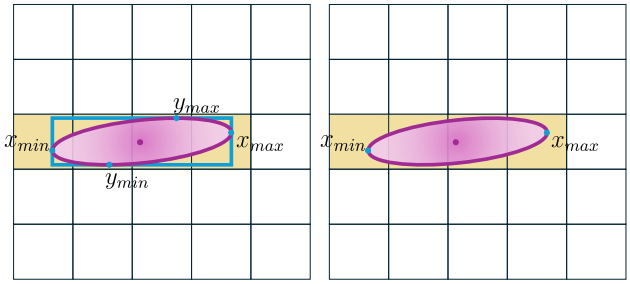

Figure 4. (Left) SnugBox and (right) AccuTile sketch of Case 1. As with Figure 2c, our AccuTile algorithm iterates over the tile rows; the only points that are processed are x_{min} and x_{max}.

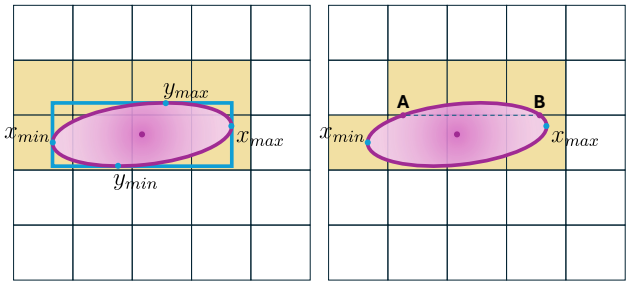

Figure 5. (Left) SnugBox and (right) AccuTile sketch of Case 2.1. Our AccuTile algorithm iterates over the tile rows; the only points that are processed are x_{min}, x_{max}, **A**, and **B**.

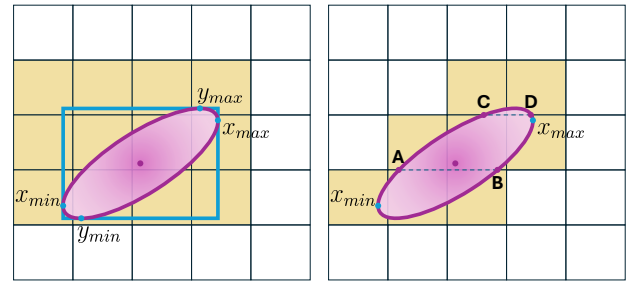

Figure 6. (Left) SnugBox and (right) AccuTile sketch of Cases 2.2, 2.3, and 3. Our AccuTile algorithm iterates over the tile rows; the only points that are processed are x_{min}, x_{max}, **A**, **B**, **C**, and **D**. A detailed walkthrough of this example is presented in Section 4.1.2.

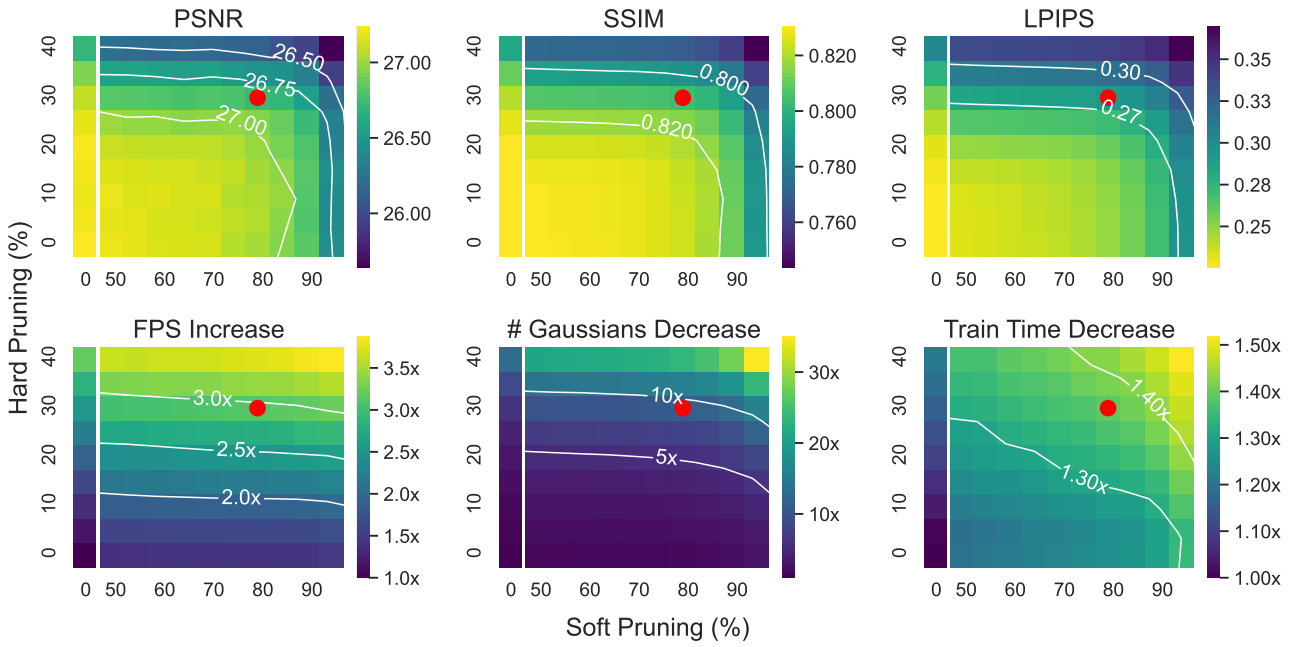

Figure 7. We sweep pruning percentages in 5% increments for Hard Pruning (0% − 40%) and Soft Pruning (0%, 50% − 95%) on all scenes listed in Section 5.1. Experiments are performed 3× on each scene without our Gaussian localization methods; the reported metrics are averaged across all runs. (0%, 0%) is the baseline 3D-GS model, the first column (0%, :) is Hard Pruning in isolation, and the first row (:, 0%) is Soft Pruning in isolation. The red dots at (80%, 30%) denote the percentage settings used in our manuscript. We report the FPS increase and the Number of Gaussians and Train Time decrease factors to be consistent with the format in Table 3.

Table 5. Average reported metrics for each pruning method across all scenes in the Tanks & Temples dataset.

Method	Comp ↑	FPS ↑	Train ↑	PSNR ↑	SSIM ↑	LPIPS ↓
3D-GS [14]	1.00×	1.00×	1.00×	23.70	0.849	0.178
Trimming [2]	4.00×	-	-	23.69	0.831	0.210
Compact [18]	2.19×	1.16×	0.76×	23.32	0.831	0.201
EAGLES [10]	-	1.73×	1.19×	23.10	0.820	0.220
Reducing [26]	2.56×	1.91×	1.27×	23.57	0.840	0.188
Light [6]	2.94×	1.97×	-	23.11	0.817	0.231
ELMGS [1]	5.00×	4.05×	-	23.90	0.825	0.233
PUP [11]	10.0×	4.00×	-	22.72	0.801	0.244
Mini-Splat [7]	9.20×	-	-	23.18	0.835	0.202
+SnugBox	0.99×	1.61×	1.11×	23.69	0.849	0.178
+AccuTile	0.99×	1.67×	1.12×	23.73	0.849	0.177
+Soft Pruning	1.69×	2.48×	1.36×	23.54	0.841	0.201
+Hard Pruning	10.1×	6.30×	1.58×	23.45	0.821	0.241

Table 6. Average reported metrics for each pruning method across all scenes in the Deep Blending dataset.

Method	Comp ↑	FPS ↑	Train ↑	PSNR ↑	SSIM ↑	LPIPS ↓
3D-GS [14]	1.00×	1.00×	1.00×	29.09	0.886	0.288
Trimming [2]	1.33×	-	-	29.43	0.897	0.267
Compact [18]	2.65×	1.37×	0.79×	29.79	0.901	0.258
EAGLES [10]	-	1.30×	1.31×	29.92	0.900	0.250
Reducing [26]	2.86×	1.79×	1.27×	29.63	0.902	0.249
Light [6]	-	-	-	-	-	-
ELMGS [1]	5.00×	4.15×	-	29.24	0.894	0.273
PUP [11]	10.0×	4.51×	-	28.85	0.881	0.301
Mini-Splat [7]	8.06×	-	-	29.98	0.908	0.253
+SnugBox	0.97×	2.11×	1.12×	29.18	0.886	0.287
+AccuTile	0.97×	2.32×	1.13×	29.12	0.885	0.288
+Soft Pruning	1.86×	3.56×	1.41×	29.29	0.889	0.296
+Hard Pruning	11.1×	7.46×	1.57×	29.32	0.887	0.311

A.2. Overall Pruning Percent Metrics

In Figure 7, we perform a parameter sweep over Hard Pruning percentages from 0% − 40% at 5% intervals and Soft Pruning percentages at 0% and from 50 − 95% at 5% intervals. We conduct each experiment 3× on each scene listed in Section 5.1 to reduce variance, then average the metrics across all runs. All experiments are run without our Gaussian localization methods – SnugBox and AccuTile – to ablate the effect of each pruning method in isolation. Our (80%, 30%) pruning percentages are empirically selected to produce a favorable balance between speed and quality.

A.3. Additional Datasets Evaluation

Table 5 and Table 6 present the average reported metrics for each pruning method across all scenes in the Tanks & Temples and Deep Blending datasets, respectively. The *Comp* column reports model size compression in terms of Gaussian count, *FPS* reports rendering speed-up, and *Train* reports training time speed-up, all with respect to the baseline 3D-GS model. PSNR, SSIM, and LPIPS are also recorded. The best and second best value for each metric are color coded; lossless methods are underlined.

Table 7. Average execution time (milliseconds) of each function across all scenes. This experiment ablates the StopThePop [27] Tile-Based Culling method with warp-level load balancing against our AccuTile algorithm. For each method, execution times are averaged over three runs, with each run rendering the test set 20 times to reduce variance. The fastest times are highlighted. Our AccuTile algorithm outperforms the Tile-Based Culling method in overall runtime by a notable margin. For detailed analysis, see Section A.4.

Method	Preprocess	Inclusive Sum	Duplicate with Keys	Radix Sort	Identify Tile Ranges	Render	Overall
Baseline	0.665	0.046	0.568	1.551	0.083	4.469	7.457
Tile-Based Culling [27]	0.811 (0.820x)	0.046	0.450 (1.263x)	0.609 (2.548x)	0.035 (2.341x)	2.027 (2.205x)	4.051 (1.841x)
AccuTile (Ours)	0.659	0.046	0.194 (2.931x)	0.610 (2.541x)	0.035 (2.338x)	2.042 (2.189x)	3.660 (2.038x)

Table 8. Average execution time (milliseconds) of each scene. This experiment ablates the StopThePop [27] Tile-Based Culling method with warp-level load balancing against our AccuTile algorithm by breaking out the per-scene execution times, which were averaged in the Overall column of Table 7. The fastest times are highlighted. Our AccuTile algorithm outperforms Tile-Based Culling on all scenes.

	Mip-NeRF 360									Tanks & Temples		Deep Blending	
Method	bicycle	bonsai	counter	flowers	garden	kitchen	room	stump	treehill	train	truck	drjohnson	playroom
Baseline	14.034	4.977	7.025	7.914	6.103	11.025	8.562	5.776	7.088	6.993	4.872	7.253	5.322
Tile-Based Culling [27]	6.609	2.674	3.261	3.474	3.888	6.994	4.741	2.997	3.058	4.417	2.860	4.131	3.554
AccuTile (Ours)	5.880	2.401	2.989	2.933	3.478	6.433	4.490	2.578	2.726	3.965	2.729	3.671	3.300

A.4. StopThePop Tile-Based Culling Ablation

We ablate our AccuTile algorithm against the StopThePop [27] Tile-Based Culling method in Tables 7 and 8. Tile-Based Culling computes a precise Gaussian-to-Tile mapping in two steps: (1) Similar to our SnugBox method, a tight, opacity-aware bounding box is computed per Gaussian; however, due to the use of thresholds in their code, not all bounding boxes are tight. (2) Each tile touching the bounding box is iteratively examined to determine if it should be included in the final Gaussian-to-Tile mapping; warp-level load balancing is used to accelerate this process.

For this ablation, we update the 3D-GS rasterizer with the Tile-Based Culling code to isolate its runtime speedup. All warp-level load balancing code is included to ensure that we compare against the most optimized version of the method. As noted in the StopThePop codebase, a padded alpha threshold is required to accurately compute bounding boxes, which, by extension, prevents undercounting Gaussian-to-Tile mappings. No padded alpha threshold is provided, so we perform this ablation without it. To ensure a fair comparison, we train three models for each scene and measure execution times with the baseline 3D-GS, Tile-Based Culling, and AccuTile renderers on each one.

Table 7 shows that our AccuTile method significantly outperforms Tile-Based Culling on Preprocess and Duplicate with Keys. Since Tile-Based Culling iterates over all candidate tiles while AccuTile does not, it requires more computation and induces a markedly higher runtime cost even with warp-level load-balancing. Surprisingly, Tile-Based Culling slightly outperforms AccuTile in the downstream functions Radix Sort, Identify Tile Ranges, and Render. However, we observe that this is caused by the aforementioned under-counting of Gaussian-to-Tile mappings; this marginal improvement disappears when padded alpha thresholds are introduced, further slowing down Preprocess and Duplicate with Keys. Additionally, as reported by Table 8, our AccuTile method consistently outperforms Tile-Based Culling across all scenes.

A.5. Per-Scene Metrics

PSNR, SSIM, LPIPS, FPS, and training times for each scene from the Mip-NeRF 360, Tanks&Temples, and Deep Blending datasets that was used in 3D-GS [14] are recorded in Tables 9, 10, 11, 12, and Table 13, respectively. The operation in each row is applied **cumulatively** to all of the following rows.

Table 9. PSNR ↑ on each scene after cumulatively applying each function.

Method	Mip-NeRF 360									Tanks & Temples		Deep Blending	
	bicycle	bonsai	counter	flowers	garden	kitchen	room	stump	treehill	train	truck	drjohnson	playroom
Baseline	25.10	32.42	29.14	21.41	27.31	31.49	31.66	26.78	22.62	22.01	25.40	28.18	30.00
+SnugBox	25.12	32.36	29.09	21.45	27.31	31.61	31.70	26.78	22.54	21.97	25.41	28.27	30.09
+AccuTile	25.13	32.42	29.13	21.43	27.33	31.65	31.68	26.80	22.58	22.00	25.45	28.23	30.00
+Soft Pruning	25.09	31.91	28.74	21.35	27.16	30.83	31.32	26.88	22.57	21.74	25.34	28.44	30.14
+Hard Pruning	24.78	31.29	28.28	21.21	26.70	29.91	30.99	26.79	22.51	21.71	25.20	28.50	30.14

Table 10. SSIM ↑ on each scene after cumulatively applying each function.

Method	Mip-NeRF 360									Tanks & Temples		Deep Blending	
	bicycle	bonsai	counter	flowers	garden	kitchen	room	stump	treehill	train	truck	drjohnson	playroom
Baseline	0.747	0.948	0.916	0.589	0.857	0.933	0.927	0.770	0.636	0.815	0.883	0.880	0.891
+SnugBox	0.749	0.948	0.916	0.591	0.857	0.933	0.928	0.771	0.636	0.815	0.883	0.880	0.892
+AccuTile	0.749	0.948	0.916	0.590	0.857	0.933	0.927	0.771	0.637	0.816	0.883	0.879	0.891
+Soft Pruning	0.741	0.941	0.904	0.582	0.848	0.921	0.920	0.776	0.630	0.803	0.878	0.884	0.893
+Hard Pruning	0.704	0.927	0.878	0.561	0.815	0.894	0.905	0.765	0.590	0.773	0.868	0.882	0.892

Table 11. LPIPS ↓ on each scene after cumulatively applying each function.

Method	Mip-NeRF 360									Tanks & Temples		Deep Blending	
	bicycle	bonsai	counter	flowers	garden	kitchen	room	stump	treehill	train	truck	drjohnson	playroom
Baseline	0.244	0.183	0.185	0.359	0.122	0.118	0.200	0.242	0.346	0.208	0.147	0.291	0.284
+SnugBox	0.241	0.183	0.185	0.358	0.122	0.117	0.199	0.241	0.345	0.208	0.147	0.291	0.284
+AccuTile	0.242	0.183	0.185	0.359	0.122	0.117	0.199	0.241	0.344	0.207	0.147	0.292	0.284
+Soft Pruning	0.271	0.197	0.212	0.379	0.147	0.141	0.222	0.258	0.390	0.237	0.165	0.297	0.295
+Hard Pruning	0.333	0.231	0.260	0.419	0.213	0.198	0.260	0.288	0.463	0.291	0.191	0.313	0.308

Table 12. FPS ↑ on each scene after cumulatively applying each function. Speed-ups ↑ are recorded in (parentheses).

Method	Mip-NeRF 360									Tanks & Temples		Deep Blending	
	bicycle	bonsai	counter	flowers	garden	kitchen	room	stump	treehill	train	truck	drjohnson	playroom
Baseline	71	201	142	126	164	91	117	172	140	141	200	138	185
+SnugBox	154 (2.15×)	358 (1.78×)	276 (1.95×)	301 (2.39×)	267 (1.62×)	146 (1.60×)	197 (1.68×)	335 (1.95×)	301 (2.15×)	228 (1.61×)	320 (1.60×)	247 (1.79×)	282 (1.53×)
+AccuTile	168 (2.35×)	413 (2.05×)	330 (2.33×)	332 (2.64×)	285 (1.73×)	155 (1.70×)	221 (1.89×)	378 (2.20×)	315 (2.25×)	248 (1.75×)	343 (1.71×)	272 (1.97×)	294 (1.59×)
+Soft Pruning	241 (3.37×)	601 (2.99×)	505 (3.56×)	497 (3.95×)	419 (2.55×)	255 (2.80×)	425 (3.63×)	612 (3.56×)	549 (3.92×)	379 (2.68×)	518 (2.59×)	423 (3.06×)	477 (2.58×)
+Hard Pruning	662 (9.25×)	978 (4.87×)	842 (5.94×)	1122 (8.91×)	825 (5.02×)	640 (7.03×)	809 (6.90×)	1277 (7.42×)	942 (6.73×)	724 (5.12×)	1392 (6.95×)	957 (6.93×)	1149 (6.21×)

Table 13. Training time ↓ in minutes on each scene after cumulatively applying each function. Speed-ups ↑ are recorded in (parentheses).

Method	Mip-NeRF 360									Tanks & Temples		Deep Blending	
	bicycle	bonsai	counter	flowers	garden	kitchen	room	stump	treehill	train	truck	drjohnson	playroom
Baseline	31.9	20.4	24.1	24.1	32.3	27.8	23.7	24.1	24.2	11.1	13.4	24.8	19.5
+SnugBox	28.2 (1.13×)	19.2 (1.07×)	21.8 (1.11×)	22.7 (1.06×)	29.9 (1.08×)	25.8 (1.08×)	21.4 (1.11×)	22.9 (1.05×)	22.4 (1.08×)	9.8 (1.13×)	12.3 (1.09×)	21.7 (1.14×)	17.8 (1.09×)
+AccuTile	27.8 (1.15×)	19.0 (1.08×)	21.3 (1.13×)	22.6 (1.07×)	29.4 (1.10×)	25.5 (1.09×)	21.1 (1.12×)	22.7 (1.06×)	22.3 (1.08×)	9.7 (1.14×)	12.2 (1.09×)	21.5 (1.15×)	17.7 (1.10×)
+Soft Pruning	23.1 (1.38×)	17.0 (1.20×)	18.6 (1.30×)	19.5 (1.23×)	23.2 (1.39×)	20.3 (1.37×)	18.3 (1.30×)	19.3 (1.25×)	18.9 (1.27×)	8.3 (1.33×)	9.7 (1.38×)	17.3 (1.43×)	14.2 (1.37×)
+Hard Pruning	19.7 (1.62×)	16.0 (1.28×)	17.7 (1.36×)	17.5 (1.38×)	20.3 (1.59×)	18.7 (1.49×)	16.9 (1.40×)	17.1 (1.41×)	16.9 (1.43×)	7.2 (1.55×)	8.3 (1.61×)	15.3 (1.62×)	12.8 (1.52×)