SparseOIT: Improving Order-Independent Transparency 3DGS via Active Set Method

Supplementary Material

A. Details of Rasterizer with Pre-rendering

In this section, we present the details of the Rasterizer with pre-rendering as Alg. 2. The ”BAN” means blend and normalize function. The ”BAU” means blend and update function. It is worth noting that the sorting step only orders entries by tile ID and the blending process is independent of the depth order of the Gaussians.

Algorithm 2 GPU software rasterization of 3D Gaussians

Require: w, h; Gaussian means M and covariances S; colors C and opacities A; view V
Ensure: Rendered image I, pre-rendered image I^{pre}
CullGaussian(p, V)
$M', S' \leftarrow$ ScreenspaceGaussians(M, S, V)
$T \leftarrow$ CreateTiles(w, h)
$L, K \leftarrow$ DuplicateWithKeys(M', T)
SortByKeys(K, L) ▷ Sort Only in Tile
$R \leftarrow$ IdentifyTileRanges(T, K)
$I \leftarrow 0$
for all tiles t in T **do**
  for all pixels i in t **do**
    $r \leftarrow$ GetTileRange(R, t)
    $I[i] \leftarrow$ BAN$(i, L, r, K, M', S', C, A, I^{\text{pre}}[i])$
    $I^{\text{pre}}[i] \leftarrow$ BAU$(i, L, r, K, M', S', C, A, I^{\text{pre}}[i])$
return I

B. Details of Backward Propagation

In this section, we present the derivation of the backward propagation of SparseOIT. We rewrite the Eq. 7 as:

$$\begin{aligned} \mathbf{C} &= T\mathbf{c}_0 + (1-T)\,F, \\ F &= \frac{P}{Q}, \quad P = \sum_{i=1}^{N} \mathbf{c}_i \alpha_i w_i, \quad Q = \sum_{i=1}^{N} \alpha_i w_i, \end{aligned} \tag{9}$$

Therefore, we can get the gradient of $\frac{\partial \mathbf{C}}{\partial \mathbf{c}_i}$, $\frac{\partial \mathbf{C}}{\partial \alpha_i}$, and $\frac{\partial \mathbf{C}}{\partial w_i}$ as:

$$\begin{aligned} \frac{\partial \mathbf{C}}{\partial \mathbf{c}_i} &= (1-T)\,\frac{\alpha_i w_i}{Q}, \\ \frac{\partial \mathbf{C}}{\partial \alpha_i} &= \frac{T}{1-\alpha_i}\,(F - \mathbf{c}_0) + (1-T)\,\frac{w_i}{Q}\,(\mathbf{c}_i - F), \\ \frac{\partial \mathbf{C}}{\partial w_i} &= (1-T)\,\frac{\alpha_i}{Q}\,(\mathbf{c}_i - F). \end{aligned} \tag{10}$$

From the backward propagation equation, we observe that the variables of each Gaussian are mutually independent. Therefore, we can apply the backward propagation with recursive Per-Splat parallelization strategy described in Sec. 4.2, or alternatively adopt the active set method to further accelerate the computation.

C. Analysis of View Subsampling Number and Update Interval for Active Set

In this section, we analyze the effects of the number of subsampled views and the update interval on the final performance. We conduct the corresponding experiments on the truck scene from the Tanks & Temples dataset based on the SparseOIT-B variant. Tab. 3 and Tab. 4 report the experimental results for the number of subsampled views and the update interval, respectively, where I denotes the update interval and S denotes the number of subsampled views. As shown in Tab. 3, increasing the number of subsampled views consistently leads to higher computational cost, while the rendering quality remains relatively stable with only minor fluctuations. Therefore, we use 30 subsampled views in our experiments, as this setting achieves a favorable trade-off between efficiency and performance. As shown in Tab. 4, the training time generally increases as the update interval becomes longer. This is because a longer update interval slows down the reduction of active 3D Gaussians, resulting in higher computational cost during training. Therefore, we choose 500 as the update interval in most cases. For a few scenes, such as *train*, *room*, and *bonsai*, however, we set the update interval to 600, which exceeds twice the number of training views, to ensure the correct execution of the algorithm.

D. Detailed Evaluation Metrics

In this section, we further introduce SparseOIT-D, a variant that integrates the CUDA acceleration techniques described in Sec. 4.2, the active set strategy, and the densification strategy from Taming-3DGS [21]. The quantitative results are reported in Tab. 6. In addition, we report per-scene PSNR, SSIM, LPIPS, training time, and the number of 3D Gaussians. Tab. 7, Tab. 8, Tab. 9 present the PSNR, SSIM, and LPIPS metrics for each scene within Mip-NeRF 360, Tanks & Temples, and Deep Blending datasets, respectively. Tab. 10 presents the per-scene training time across the three datasets, and Tab. 11 provides the 3D Gaussian counts.

Table 3. Impact of the number of subsampled views on performance in the truck scene.

Parameters	PSNR ↑	SSIM ↑	LPIPS ↓	Time ↓	N(k)
I=500, S=10	25.48	0.8745	0.1521	464	2307
I=500, S=20	25.48	0.8753	0.1514	459	2285
I=500, S=30	25.53	0.8753	0.1515	456	2288
I=500, S=40	25.56	0.8752	0.1516	460	2279
I=500, S=50	25.52	0.8750	0.1518	462	2290
I=500, S=60	25.53	0.8753	0.1518	464	2283
I=500, S=70	25.55	0.8753	0.1512	477	2294
I=500, S=80	25.51	0.8749	0.1524	471	2282
I=500, S=90	25.48	0.8748	0.1524	470	2269
I=500, S=100	25.55	0.8755	0.1519	472	2255

Table 4. Impact of the update intervals on performance in the truck scene.

Parameters	PSNR ↑	SSIM ↑	LPIPS ↓	Time ↓	N(k)
I=500, S=30	25.53	0.8753	0.1515	456	2288
I=600, S=30	25.57	0.8755	0.1516	461	2290
I=700, S=30	25.51	0.8746	0.1516	455	2296
I=800, S=30	25.52	0.8753	0.1523	459	2261
I=900, S=30	25.52	0.8754	0.1524	459	2251
I=1000, S=30	25.54	0.8752	0.1518	464	2275
I=2000, S=30	25.52	0.8747	0.1530	480	2285
I=3000, S=30	25.52	0.8750	0.1521	492	2257
I=4000, S=30	25.51	0.8754	0.1517	503	2265
I=5000, S=30	25.58	0.8755	0.1520	516	2274

E. Further Analysis and Discussion

From our empirical observations, the outdoor scenes in the Mip-NeRF 360 dataset tend to overfit under the learning rates adopted in the main paper. To further investigate this issue, we additionally conduct experiments with the learning rate of parameter v reduced to 0.0005, while keeping all other settings unchanged. Following the experimental setting in Sec. D, we further provide both the average results on the Mip-NeRF 360 dataset in Tab. 5 and the detailed per-scene results for the outdoor scenes in Tab. 12 and Tab. 13.

It can be observed that the rendering quality on the outdoor scenes of Mip-NeRF 360 improves substantially. This phenomenon may be due to the fact that the current weight formulation of 3D Gaussians is not sufficiently accurate, which also makes the optimization more difficult. Moreover, the slightly worse performance of the current OIT formulation compared with 3DGS may also stem from the relatively lower quality metrics on the outdoor scenes of the Mip-NeRF 360 dataset.

In addition, empirical results from our initial experiments suggest that the random sampling strategy applied during densification can improve novel view synthesis performance on the evaluation set across multiple scenes, with representative examples including *train*, *drjohnson*, and *treehill*. This may be attributed to the fact that a larger number of 3D Gaussians makes the scene more prone to overfitting, thereby degrading rendering quality on unseen views.

Table 5. Quantitative results on Mip-NeRF 360 under adjusted learning rate

Method	PSNR↑	SSIM↑	LPIPS↓	Time↓	N(k)
3DGS [15]	27.68	0.8214	0.1771	909	2679
Taming-3DGS [21]	27.44	0.8012	0.2193	183	665
SortFree [14]	27.33	0.8067	0.1792	2302	4314
SparseOIT-A	27.38	0.8058	0.1964	564	2082
SparseOIT-B	27.38	0.8058	0.1973	460	2094
SparseOIT-C	27.11	0.7884	0.2268	201	686
SparseOIT-D	26.92	0.7805	0.2377	175	686

Table 6. Comparison with baselines on Tanks & Temples, DeepBlending, and Mip-NeRF datasets. $N(k)$ denotes the number of Gaussians divided by 1000. Best and second-best results are highlighted in best, second best.

Method	Tanks & Temples					DeepBlending					Mip-NeRF 360				
	PSNR↑	SSIM↑	LPIPS↓	Time↓	$N(k)$	PSNR↑	SSIM↑	LPIPS↓	Time↓	$N(k)$	PSNR↑	SSIM↑	LPIPS↓	Time↓	$N(k)$
3DGS [15]	23.78	0.8494	0.1704	705	1569	29.70	0.9027	0.2409	1213	2459	27.68	0.8214	0.1771	909	2679
Taming-3DGS [21]	23.70	0.8320	0.2122	153	319	29.70	0.8992	0.2734	156	294	27.44	0.8012	0.2193	183	665
SortFree [14]	22.97	0.8299	0.1814	2159	3765	29.76	0.9016	0.2399	2065	2843	27.33	0.8067	0.1792	2302	4314
SparseOIT-A	23.63	0.8422	0.1784	559	2055	29.76	0.9030	0.2479	355	1249	27.19	0.8019	0.2023	531	2126
SparseOIT-B	23.68	0.8429	0.1798	445	2052	29.80	0.9043	0.2486	309	1251	27.21	0.8027	0.2040	408	2121
SparseOIT-C	23.39	0.8206	0.2255	160	319	29.87	0.9010	0.2692	159	295	26.98	0.7802	0.2394	191	686
SparseOIT-D	23.34	0.8143	0.2374	159	319	29.84	0.8993	0.2783	150	295	26.84	0.7737	0.2500	170	686

Table 7. PSNR scores of our method on the Mip-NeRF 360 dataset, the Tanks & Temples dataset, and the Deep Blending dataset.

Method	Mip-NeRF 360									Tanks & Temples		Deep Blending	
	bicycle	flowers	garden	stump	treehill	room	counter	kitchen	bonsai	truck	train	drjohnson	playroom
3DGS [15]	25.24	21.50	27.37	26.64	22.54	32.23	29.37	31.80	32.47	25.44	22.12	29.41	30.00
Taming-3DGS [21]	24.85	21.01	27.25	25.99	22.89	32.14	28.91	31.65	32.26	25.17	22.24	29.43	29.98
SortFree [14]	23.89	20.58	27.07	24.75	21.09	32.90	30.30	31.90	33.45	24.58	21.36	29.51	30.00
SparseOIT-A	23.92	20.42	26.95	24.85	22.32	32.11	29.88	31.31	32.93	25.45	21.82	29.59	29.93
SparseOIT-B	23.86	20.49	27.11	25.02	22.34	32.13	29.83	31.33	32.78	25.53	21.82	29.64	29.95
SparseOIT-C	23.53	20.11	26.95	24.64	22.40	31.87	29.36	31.21	32.74	24.95	21.85	29.56	30.18
SparseOIT-D	23.66	20.02	26.53	24.74	22.53	31.55	29.31	30.72	32.48	24.96	21.73	29.54	30.13

Table 8. SSIM scores of our method on the Mip-NeRF360 dataset, the Tanks & Temples dataset, and the Deep Blending dataset.

Method	Mip-NeRF 360									Tanks & Temples		Deep Blending	
	bicycle	flowers	garden	stump	treehill	room	counter	kitchen	bonsai	truck	train	drjohnson	playroom
3DGS [15]	0.7647	0.6029	0.8640	0.7714	0.6326	0.9415	0.9176	0.9444	0.9532	0.8810	0.8178	0.9026	0.9028
Taming-3DGS [21]	0.7169	0.5531	0.8532	0.7351	0.6227	0.9327	0.9084	0.9406	0.9485	0.8638	0.8002	0.8996	0.8987
SortFree [14]	0.7281	0.5875	0.8570	0.7054	0.6126	0.9448	0.9241	0.9444	0.9563	0.8665	0.7933	0.9012	0.9020
SparseOIT-A	0.6942	0.5743	0.8492	0.7181	0.6219	0.9430	0.9205	0.9422	0.9534	0.8753	0.8091	0.9040	0.9020
SparseOIT-B	0.6952	0.5751	0.8504	0.7233	0.6218	0.9429	0.9201	0.9423	0.9531	0.8753	0.8106	0.9050	0.9037
SparseOIT-C	0.6406	0.5319	0.8379	0.6822	0.5970	0.9336	0.9117	0.9375	0.9495	0.8532	0.7880	0.9006	0.9015
SparseOIT-D	0.6347	0.5189	0.8220	0.6778	0.5917	0.9299	0.9071	0.9344	0.9466	0.8527	0.7758	0.8985	0.9001

Table 9. LPIPS scores of our method on the Mip-NeRF360 dataset, the Tanks & Temples dataset, and the Deep Blending dataset.

Method	Mip-NeRF 360									Tanks & Temples		Deep Blending	
	bicycle	flowers	garden	stump	treehill	room	counter	kitchen	bonsai	truck	train	drjohnson	playroom
3DGS [15]	0.2105	0.3379	0.1078	0.2153	0.3287	0.1140	0.1143	0.0677	0.0974	0.1428	0.1980	0.2388	0.2429
Taming-3DGS [21]	0.2970	0.4088	0.1287	0.2924	0.3892	0.1366	0.1320	0.0759	0.1132	0.1854	0.2390	0.2692	0.2777
SortFree [14]	0.2352	0.3101	0.1142	0.2510	0.3304	0.1081	0.1046	0.0677	0.0916	0.1509	0.2118	0.2364	0.2433
SparseOIT-A	0.3098	0.3658	0.1332	0.2556	0.3652	0.1116	0.1110	0.0704	0.0983	0.1502	0.2067	0.2474	0.2483
SparseOIT-B	0.3135	0.3682	0.1351	0.2561	0.3697	0.1126	0.1124	0.0704	0.0978	0.1515	0.2081	0.2487	0.2485
SparseOIT-C	0.3711	0.4218	0.1559	0.3413	0.4117	0.1351	0.1301	0.0793	0.1082	0.2005	0.2505	0.2633	0.2750
SparseOIT-D	0.3829	0.4322	0.1836	0.3518	0.4245	0.1420	0.1364	0.0834	0.1132	0.2031	0.2718	0.2750	0.2817

Table 10. Training time of our method on the Mip-NeRF360 dataset, the Tanks & Temples dataset, and the Deep Blending dataset.

Method	Mip-NeRF 360									Tanks & Temples		Deep Blending	
	bicycle	flowers	garden	stump	treehill	room	counter	kitchen	bonsai	truck	train	drjohnson	playroom
3DGS [15] 3DGS [15]	1467	975	1396	1207	1082	495	494	621	443	776	634	1396	1031
Taming-3DGS [21]	206	178	324	156	199	129	144	162	145	144	161	165	147
SortFree [14]	3423	3494	3063	3251	3198	1037	1159	1129	965	2076	2242	2640	1491
SparseOIT-A	523	500	817	933	497	333	373	430	376	612	505	314	396
SparseOIT-B	417	411	603	607	409	262	297	363	303	456	434	288	329
SparseOIT-C	199	170	352	163	201	140	158	176	165	154	166	171	146
SparseOIT-D	167	164	241	158	176	134	158	175	154	157	162	159	141

Table 11. Per-scene 3D Gaussian counts of our method on the Mip-NeRF 360, Tanks & Temples, and Deep Blending datasets, where each value is reported as $N(k)$, the number of Gaussians divided by 1000.

Method	**Mip-NeRF 360**									**Tanks & Temples**		**Deep Blending**	
	bicycle	flowers	garden	stump	treehill	room	counter	kitchen	bonsai	truck	train	drjohnson	playroom
3DGS [15]	4876	2865	4069	4327	3269	1141	994	1509	1064	2054	1084	3076	1841
Taming-3DGS [21]	813	575	1903	480	785	225	311	482	413	272	365	404	185
SortFree [14]	6312	7296	5238	6498	6269	1676	2076	1872	1591	3694	3835	3650	2035
SparseOIT-A	2293	2159	3227	4204	2124	1133	1310	1414	1268	2275	1835	1118	1380
SparseOIT-B	2261	2136	3232	4191	2110	1140	1322	1426	1271	2288	1817	1118	1384
SparseOIT-C	814	575	2081	483	785	225	312	483	413	272	366	404	185
SparseOIT-D	814	575	2081	483	785	225	311	483	413	272	366	404	185

Table 12. PSNR, SSIM and LPIPS scores of our method in the outdoor scenes of Mip-NeRF 360 dataset under adjusted learning rate

Method	**PSNR**					**SSIM**					**LPIPS**				
	bicycle	flowers	garden	stump	treehill	bicycle	flowers	garden	stump	treehill	bicycle	flowers	garden	stump	treehill
3DGS [15]	25.24	21.50	27.37	26.64	22.54	0.7647	0.6029	0.8640	0.7714	0.6326	0.2105	0.3379	0.1078	0.2153	0.3287
Taming-3DGS [21]	24.85	21.01	27.25	25.99	22.89	0.7169	0.5531	0.8532	0.7351	0.6227	0.2970	0.4088	0.1287	0.2924	0.3892
SortFree [14]	23.89	20.58	27.07	24.75	21.09	0.7281	0.5875	0.8570	0.7054	0.6126	0.2352	0.3101	0.1142	0.2510	0.3304
SparseOIT-A	24.25	20.72	27.13	25.42	22.63	0.7239	0.5635	0.8573	0.7247	0.6239	0.2705	0.3734	0.1168	0.2521	0.3638
SparseOIT-B	24.35	20.78	27.13	25.44	22.65	0.7244	0.5629	0.8570	0.7268	0.6229	0.2720	0.3740	0.1182	0.2518	0.3665
SparseOIT-C	23.85	20.19	26.95	25.11	22.72	0.6870	0.5199	0.8490	0.6992	0.6080	0.3218	0.4233	0.1304	0.3167	0.3960
SparseOIT-D	23.72	20.10	26.75	25.11	22.55	0.6748	0.5028	0.8408	0.6914	0.5968	0.3389	0.4376	0.1437	0.3301	0.4142

Table 13. Training time and per-scene 3D Gaussian counts of our method in the outdoor scenes of Mip-NeRF 360 dataset under adjusted learning rate

Method	**Time**					**N(k)**				
	bicycle	flowers	garden	stump	treehill	bicycle	flowers	garden	stump	treehill
3DGS [15]	1467	975	1396	1207	1082	4876	2865	4069	4327	3269
Taming-3DGS [21]	206	178	324	156	199	813	575	1903	480	785
SortFree [14]	3423	3494	3063	3251	3198	6312	7296	5238	6498	6269
SparseOIT-A	574	585	908	980	513	2345	2198	3206	3821	2041
SparseOIT-B	493	491	731	735	461	2349	2177	3212	3904	2043
SparseOIT-C	220	187	383	171	213	814	575	2081	484	785
SparseOIT-D	177	169	264	156	188	814	575	2081	484	785