3D Student Splatting and Scooping - Supplementary Material

Jialin Zhu[1], Jiangbei Yue[2], Feixiang He[1], He Wang [1,3*]
[1]University College London, UK [2]University of Leeds, UK
[3]AI Centre, University College London, UK

1. Additional Experimental Results

We show more visual results of general benchmarks for Student Splatting and Scooping (SSS) against selected baselines: 3D Gaussian Splatting (3DGS) [7], 3D Half-Gaussian Splatting (3DHGS) [10], Generalized Exponential Splattin (GES) [4] and 3D Gaussian Splatting as Markov Chain Monte Carlo (3DGS-MCMC) [8] in Fig. 1. For (a) in Fig. 1, there are a few food residues in the metal bowl. Only 3DGS-MCMC and SSS can successfully restore these details. Furthermore, SSS also achieves a better reconstruction of colors than 3DGS-MCMC with these residues. SSS is the only method that can reproduce the sharp details of the chair refracted in the transparent water glass in Fig. 1 (b). For (c) in Fig. 1, the difficulty lies in reconstructing the details of the upper wall edge and the items on the cabinet (texts, etc.). Considering these two difficulties, SSS performs best in this scenario. SSS can also ensure that the details at the edge of the image are restored to the greatest extent, which is reflected in (d) and (e) of Fig. 1. In addition, SSS is also the best for reconstructing pure color areas (sky, wall, etc.).

We have illustrated the results of the Train scene in the Tanks & Temples dataset [9] in the main context for the varying component numbers experiment. Here we further show two more scenes (room from Mip-MeRF 360 [2] and drjohnson from Deep Blending [6]) in Figs. 2 and 3. In Fig. 2, the difficulty of 3D reconstruction lies in the texture of the carpet. Both 3DGS, 3DHGS, and GES can only restore the approximate shape but cannot take into account the details when the number of components is small. 3DGS-MCMC and SSS can restore the details of the carpet. However, because SSS has a scooping operation, the carpet it reconstructs has a more realistic texture. 3DGS, 3DHGS, and GES are completely unable to reconstruct quality results in the setting of 80k to 208k components in Fig. 3. The reconstruction of 3DGS-MCMC is good enough but loses some details (window gaps inside the basket marked area), while SSS is still the best method for capturing details.

*Corresponding author, he_wang@ucl.ac.uk

Method \ Scene	bicycle	bonsai	counter	garden	kitchen	room	stump	average
3DGS	25.25	31.98	28.70	27.41	30.32	30.63	26.55	28.69
Mip-NeRF	24.37	33.46	29.55	26.98	32.23	31.64	26.40	29.23
Scaffold-GS	24.50	32.70	29.34	27.17	31.30	31.93	26.27	28.84
3DHGS	25.39	33.30	29.62	27.68	32.17	32.12	26.64	29.56
3DGS-MCMC	26.15	32.88	29.51	28.16	32.27	32.48	27.80	29.89
SSS	25.68	33.50	29.87	28.09	32.43	32.57	27.17	29.90

Table 1. **PSNR results for every scene in Mip-NeRF 360 dataset.** The red, orange, and yellow colors represent the top three results. Competing metrics are extracted from respective papers, and ours are reported as the average of three runs.

1.1. Detailed Results on Each Scene

We show detailed comparisons between our method and the baselines on every scene on Peak Signal-to-Noise Ratio (PSNR), Structural Similarity Index Metric (SSIM) [14], and Learned Perceptual Image Patch Similarity (LPIPS) [16] metrics among three datasets (Mip-NeRF 360 [2], Tanks & Temples [9], Deep Blending [6]) in Tabs. 1 to 6.

Note we only include baselines that provide detailed evaluation scores on each scene. These baselines are Mip-NeRF [1], 3DGS [7], Scaffold-GS [11], 3DHGS [10] and 3DGS-MCMC [8]. This is due to the intrinsic randomness in the training of these methods. When we re-train the models ourselves and often obtain slightly different results from their papers. Therefore, we use the results reported in their original papers.

Overall, our method outperforms all the baselines. A close second is 3DGS-MCMC, which is a state-of-the-art model recently. There are some baselines that outperform both our method and 3DGS-MCMC on individual scenes under some metrics, but overall, our method and 3DGS-MCMC are the best and second best methods.

Beyond the selected methods for baseline, we do realize that there might be other methods that are not included here but achieve higher scores on certain scenes and metrics, especially when it comes to specific application settings, *e.g.* 3D reconstruction. However, our goal here is to restrict our comparison to the methods that aim to improve 3DGS on its fundamental formulation and can be potentially used as a generic-purposed component.

(a) Counter from Mip-NeRF 360 [2] (b) Kitchen from Mip-NeRF 360 [2] (c) Room from Mip-NeRF 360 [2] (d) Train from Tanks & Temples [9] (e) Playroom from Deep Blending [6]

Figure 1. **Visual comparison.** (a) SSS restores the best details inside the metal bowl. (b) SSS is the only one that can reconstruct the details of the chair refracted in the transparent cup. (c) The reconstruction of the wall edge (bright blue box) and the font were both done best by SSS. (d) SSS's details on the distant woods and the reconstruction of the sky are the best. (e) The reconstruction of the pattern on the wall is SSS at its best.

Method \ Scene	bicycle	bonsai	counter	garden	kitchen	room	stump	average
3DGS	0.771	0.938	0.905	0.868	0.922	0.914	0.775	0.870
Mip-NeRF	0.685	0.941	0.894	0.813	0.920	0.913	0.744	0.844
Scaffold-GS	0.705	0.946	0.914	0.842	0.928	0.925	0.784	0.848
3DHGS	0.768	0.950	0.909	0.868	0.930	0.921	0.770	0.873
3DGS-MCMC	0.810	0.950	0.920	0.890	0.940	0.940	0.820	0.900
SSS	0.798	0.956	0.926	0.882	0.939	0.938	0.813	0.893

Table 2. **SSIM results for every scene in Mip-NeRF 360 dataset.** The red, orange, and yellow colors represent the top three results. Competing metrics are extracted from respective papers, and ours are reported as the average of three runs.

Method \ Scene	bicycle	bonsai	counter	garden	kitchen	room	stump	average
3DGS	0.205	0.205	0.204	0.103	0.129	0.220	0.210	0.182
Mip-NeRF	0.301	0.176	0.204	0.170	0.127	0.211	0.261	0.207
Scaffold-GS	0.306	0.185	0.191	0.146	0.126	0.202	0.284	0.220
3DHGS	0.202	0.180	0.201	0.104	0.125	0.220	0.215	0.178
3DGS-MCMC	0.180	0.220	0.220	0.100	0.140	0.250	0.190	0.190
SSS	0.173	0.151	0.156	0.009	0.104	0.167	0.174	0.145

Table 3. **LPIPS results for every scene in Mip-NeRF 360 dataset.** The red, orange, and yellow colors represent the top three results. Competing metrics are extracted from respective papers, and ours are reported as the average of three runs.

1.2. Detailed Results on Varying Component Numbers

We show detailed results of varying component numbers, on every scene with PSNR, SSIM, and LPIPS in Mip-NeRF 360, Tanks & Temples, and Deep Blending datasets. Since we need to re-train all methods for comparison, we ensure the implementation is as fair as possible. Based on whether codes are open sourced and furthermore how easily they can be adapted for comparison (explained later), we selected

Dataset - Scene	Tanks&Temples			Deep Blending		
Method	train	truck	average	drjohnson	playroom	average
3DGS	21.09	25.18	23.14	28.77	30.04	29.41
Mip-NeRF	19.52	24.91	22.22	29.14	29.66	29.40
Scaffold-GS	22.15	25.77	23.96	29.80	30.62	30.21
3DHGS	22.95	26.04	24.49	29.32	30.20	29.76
3DGS-MCMC	22.47	26.11	24.29	29.00	30.33	29.67
SSS	23.32	26.41	24.87	29.66	30.47	30.07

Table 4. **PSNR results for every scene in Tanks & Temples and Deep Blending dataset.** The red, orange, and yellow colors represent the top three results. Competing metrics are extracted from respective papers, and ours are reported as the average of three runs.

Dataset - Scene	Tanks&Temples			Deep Blending		
Method	train	truck	average	drjohnson	playroom	average
3DGS	0.802	0.879	0.841	0.899	0.906	0.903
Mip-NeRF	0.660	0.857	0.759	0.901	0.900	0.901
Scaffold-GS	0.822	0.883	0.853	0.907	0.904	0.906
3DHGS	0.827	0.887	0.857	0.904	0.907	0.905
3DGS-MCMC	0.830	0.890	0.860	0.890	0.900	0.890
SSS	0.850	0.897	0.873	0.905	0.909	0.907

Table 5. **SSIM results for every scene in Tanks & Temples and Deep Blending dataset.** The red, orange, and yellow colors represent the top three results. Competing metrics are extracted from respective papers, and ours are reported as the average of three runs.

Dataset - Scene	Tanks&Temples			Deep Blending		
Method	train	truck	drjohnson	playroom		
3DGS	0.218	0.148	0.183	0.244	0.241	0.243
Mip-NeRF	0.354	0.159	0.257	0.237	0.252	0.245
Scaffold-GS	0.206	0.147	0.177	0.250	0.258	0.254
3DHGS	0.197	0.141	0.169	0.240	0.243	0.242
3DGS-MCMC	0.240	0.140	0.860	0.330	0.310	0.320
SSS	0.166	0.109	0.138	0.249	0.245	0.247

Table 6. **LPIPS results for every scene in Tanks & Temples and Deep Blending dataset.** The red, orange, and yellow colors represent the top three results. Competing metrics are extracted from respective papers, and ours are reported as the average of three runs.

3DGS [7], GES [4], 3DHGS [10] and 3DGS-MCMC [8] as the baselines. We use δ to represent the number of the initial components from Structure from Motion (SFM) reconstruction [13] for different scenes.

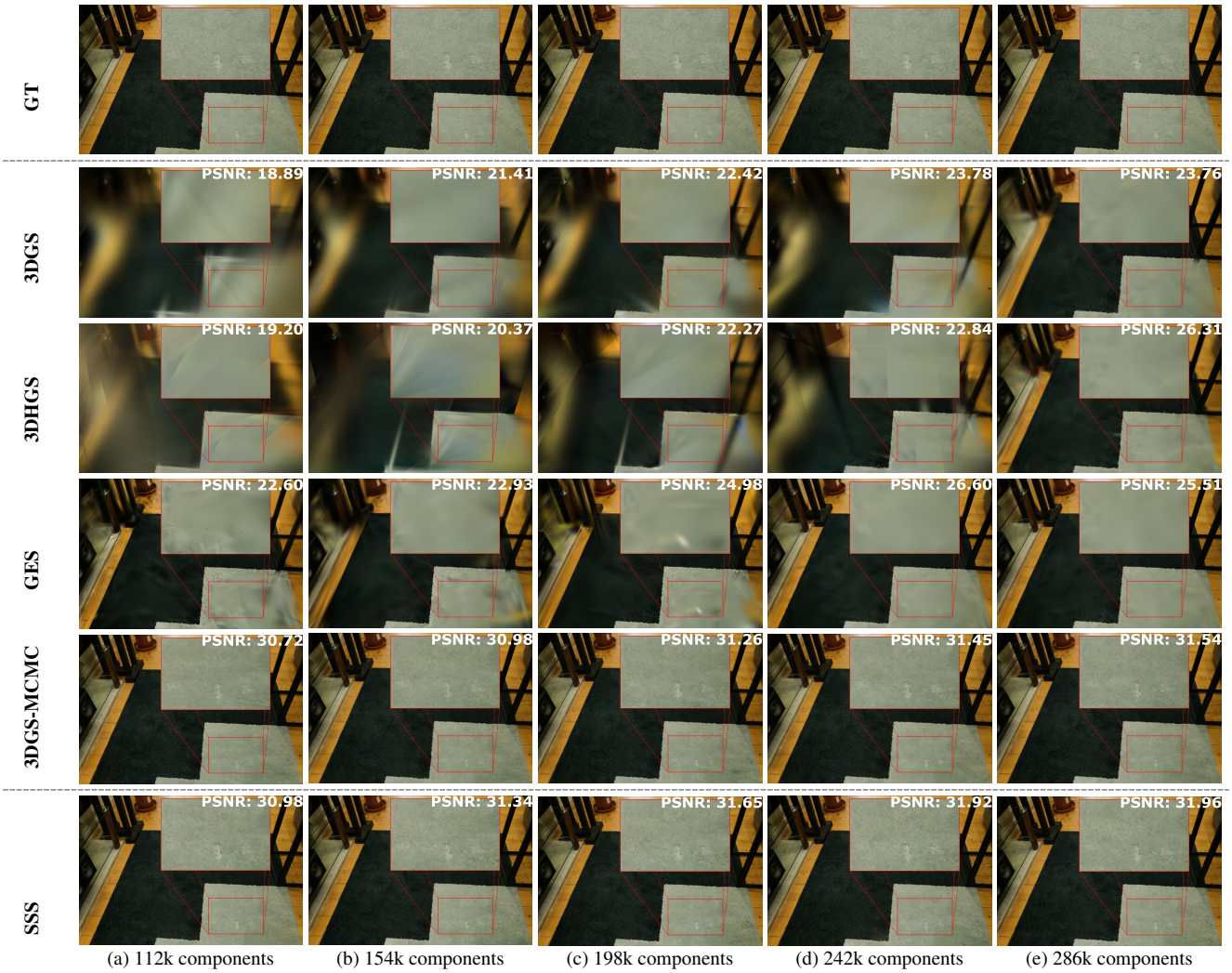
The results are shown in Tabs. 7 to 12. In total, there are 11 (scenes) $\times$ 5 (component numbers) $\times$ 3 (metrics) = 165 comparisons. For the absolute majority, SSS achieves the best. Furthermore, when it is not the best method, it is the close second to 3DGS-MCMC. This is an exhaustive comparison of many datasets, metrics, and more importantly different numbers of components.

1.3. Ablation Study

We conduct an ablation study to show the effectiveness of various components in SSS. We report the results on one dataset in Tab. 13. Universally, by only replacing Gaus-

Scene - Components	Mip-NeRF 360 - average				
Method	δ	1.4δ	1.8δ	2.2δ	2.6δ
3DGS	19.16	20.32	20.50	21.22	21.75
GES	20.59	21.60	22.78	23.52	23.86
3DHGS	19.63	20.33	20.39	20.99	22.12
3DGS-MCMC	27.47	27.89	28.18	28.39	28.56
SSS	27.72	28.14	28.43	28.66	28.84
Scene - Components	Mip-NeRF 360 - bicycle				
Method	54k	75k	97k	118k	140k
3DGS	18.32	18.53	18.62	18.75	18.52
GES	18.31	18.29	18.20	18.13	18.19
3DHGS	18.45	18.72	18.62	18.61	18.50
3DGS-MCMC	23.04	23.42	23.74	23.97	24.17
SSS	22.97	23.29	23.53	23.71	23.95
Scene - Components	Mip-NeRF 360 - bonsai				
Method	206k	280k	360k	440k	520k
3DGS	20.50	22.93	23.39	23.97	24.54
GES	22.47	24.12	25.51	27.02	29.27
3DHGS	21.37	22.90	23.45	24.62	24.82
3DGS-MCMC	31.14	31.56	31.85	32.03	32.17
SSS	31.67	32.21	32.52	32.75	32.94
Scene - Components	Mip-NeRF 360 - counter				
Method	155k	224k	288k	352k	416k
3DGS	18.26	19.07	18.84	20.78	23.73
GES	19.68	23.64	26.35	26.74	27.45
3DHGS	17.77	18.36	18.89	20.48	23.59
3DGS-MCMC	28.38	28.68	28.82	28.93	29.03
SSS	28.71	29.06	29.21	29.39	29.48
Scene - Components	Mip-NeRF 360 - garden				
Method	138k	196k	252k	308k	364k
3DGS	16.73	17.34	17.49	17.81	18.05
GES	16.62	16.94	17.24	17.58	17.92
3DHGS	17.36	17.44	17.70	18.25	18.49
3DGS-MCMC	25.23	25.70	26.02	26.26	26.47
SSS	25.32	25.80	26.17	26.41	26.64
Scene - Components	Mip-NeRF 360 - kitchen				
Method	241k	336k	432k	528k	624k
3DGS	21.19	22.70	22.10	22.63	22.70
GES	24.78	25.74	27.65	29.34	29.54
3DHGS	23.19	24.20	21.40	21.61	22.60
3DGS-MCMC	30.11	30.78	31.05	31.26	31.45
SSS	30.84	31.33	31.64	31.77	31.97
Scene - Components	Mip-NeRF 360 - room				
Method	112k	154k	198k	242k	286k
3DGS	18.89	21.41	22.42	23.78	23.76
GES	22.60	22.93	24.98	26.60	25.51
3DHGS	19.20	20.37	22.27	22.83	26.31
3DGS-MCMC	30.72	30.98	31.26	31.45	31.54
SSS	30.98	31.34	31.65	31.92	31.96
Scene - Components	Mip-NeRF 360 - stump				
Method	32k	44.8k	57.6k	70.4k	83.2k
3DGS	20.21	20.26	20.61	20.82	20.92
GES	19.67	19.55	19.51	19.25	19.13
3DHGS	20.08	20.31	20.42	20.52	20.53
3DGS-MCMC	23.64	24.08	24.51	24.84	25.09
SSS	23.57	23.95	24.31	24.66	24.91

Table 7. **PSNR results of varying component numbers experiments for every scene in the Mip-NeRF 360 dataset.**



(a) 112k components (b) 154k components (c) 198k components (d) 242k components (e) 286k components

Figure 2. **Visual results of all methods with varying component numbers of room scene from Mip-NeRF 360.** Only 3DGS-MCMC and SSS can restore the details of the carpet, but the result of SSS obviously has a more realistic carpet texture.

sians with Student's t distributions (SGD+t-distribution), it already outperforms Mip-NeRF, 3DGS, and GES, demonstrating improved expressivity. Further with SGHMC, it is already the best method, showing the advantage of the proposed sampling and the importance of a good sampler in the optimization process. Finally, adding negative components further improves the results.

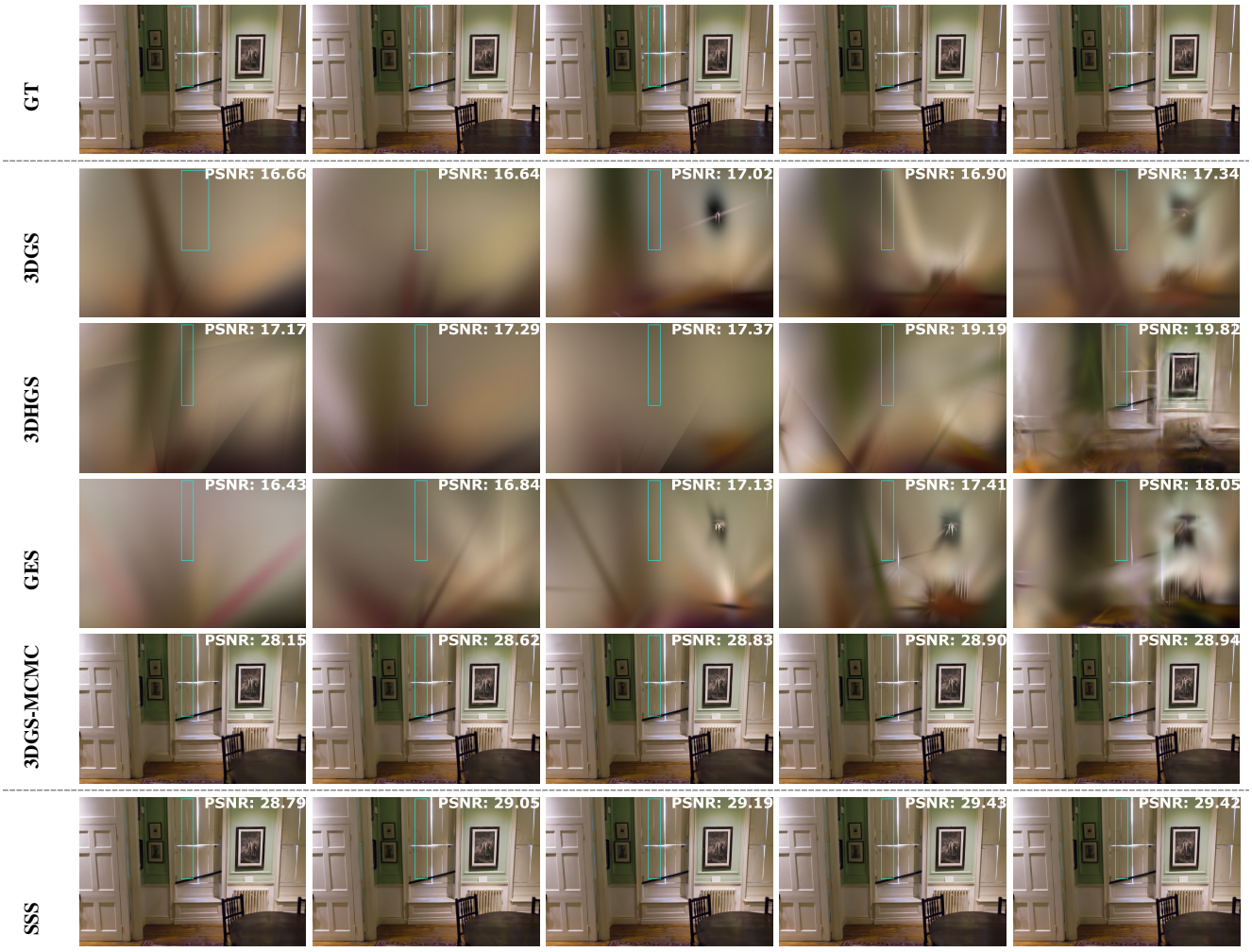
The detailed results of Ablation Study (effect of each contribution in SSS) on every scene with PSNR, SSIM, and LPIPS metrics among Mip-NeRF 360, Tanks & Temples and Deep Blending datasets are in Tabs. 14 to 19.

More ablation We also show comparison of applying SGHMC with vanilla 3DGS and positive t-distributions only with Tanks & Temples and Deep Blending datasets to the ablations (Tab. 20). Replacing SGD with SGHMC already improves the results. Replacing Gaussians with positive t-distributions further improves the PSNR but slightly reduces SSIM and LPIPS. Nonetheless, our Full model is obviously the best. While individual techniques alone might provide merely small improvements, SSS as a whole is the SOTA.

2. Sampling Effects on Learning

When trying the original SGD with only positive Student's t components, the learned ν values were not ideal. Considering Gaussian is simply a Student's t distribution with fixed $\nu = \infty$, it shows that ν introduces undesirable local minima. This was mitigated by SGHMC as the friction term decouples parameters, but the sampling became slow compared to vanilla 3DGS.

We compare the learned ν distributions between



(a) 80k components (b) 112k components (c) 144k components (d) 176k components (e) 208k components

Figure 3. **Visual results of all methods with varying component numbers of drjohnson scene from Deep Blending.** 3DGS, 3DHSGS, and GES are completely unable to reconstruct quality results. The results of 3DGS-MCMC and SSS are relatively better. SSS can restore more details (such as the window gaps in the blue box) than 3DGS-MCMC.

SGHMC (decoupling) and the 3DGS optimization (no decoupling) in Fig. 4. SGHMC learned a distribution across a wide range, with no mode collapse and fully utilizing the representation power of t-distribution. 3DGS optimization learns a distribution heavily concentrated in some areas (near 1, likely mode collapse), unable to explore the full space of t-distribution.

3. Implementation Details

The implementation of SSS is based on open-source codes of the vanilla 3D Gaussian Splatting (3DGS) [7] and 3DGS-MCMC [8]. We modified various parts to replace Gaussians with Student's t distributions for splatting algorithm [17, 18]. These modifications are adapted to both forward and backward procedures. We show the formulae of the forward

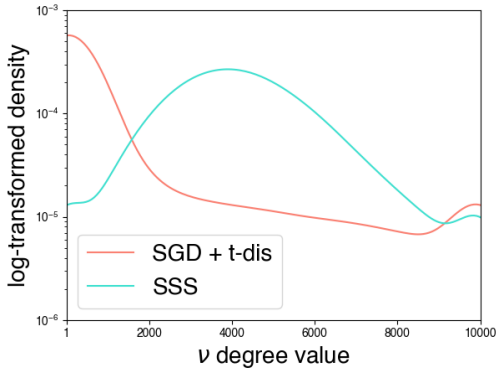


Figure 4. **Sampling effects on learning** .

Scene - Components	Mip-NeRF 360 - average				
Method	δ	1.4δ	1.8δ	2.2δ	2.6δ
3DGS	0.584	0.609	0.610	0.626	0.640
GES	0.612	0.630	0.652	0.667	0.674
3DHGS	0.591	0.603	0.599	0.611	0.634
3DGS-MCMC	0.803	0.818	0.829	0.837	0.844
SSS	0.802	0.816	0.827	0.835	0.842
Scene - Components	Mip-NeRF 360 - bicycle				
Method	54k	75k	97k	118k	140k
3DGS	0.367	0.375	0.377	0.382	0.377
GES	0.372	0.369	0.368	0.368	0.370
3DHGS	0.369	0.374	0.372	0.376	0.373
3DGS-MCMC	0.603	0.630	0.650	0.666	0.679
SSS	0.597	0.620	0.639	0.652	0.666
Scene - Components	Mip-NeRF 360 - bonsai				
Method	206k	280k	360k	440k	520k
3DGS	0.790	0.822	0.820	0.828	0.838
GES	0.811	0.836	0.861	0.882	0.911
3DHGS	0.802	0.821	0.827	0.837	0.839
3DGS-MCMC	0.938	0.942	0.945	0.947	0.948
SSS	0.938	0.944	0.948	0.950	0.951
Scene - Components	Mip-NeRF 360 - counter				
Method	155k	224k	288k	352k	416k
3DGS	0.663	0.686	0.682	0.731	0.799
GES	0.696	0.785	0.854	0.866	0.882
3DHGS	0.650	0.667	0.680	0.719	0.788
3DGS-MCMC	0.901	0.907	0.911	0.914	0.916
SSS	0.898	0.906	0.911	0.915	0.917
Scene - Components	Mip-NeRF 360 - garden				
Method	138k	196k0	252k	308k	364k
3DGS	0.366	0.386	0.399	0.412	0.424
GES	0.373	0.390	0.402	0.416	0.430
3DHGS	0.383	0.394	0.410	0.423	0.433
3DGS-MCMC	0.757	0.784	0.802	0.814	0.823
SSS	0.758	0.785	0.803	0.815	0.825
Scene - Components	Mip-NeRF 360 - kitchen				
Method	241k	336k	432k	528k	624k
3DGS	0.762	0.796	0.760	0.769	0.776
GES	0.856	0.838	0.875	0.908	0.912
3DHGS	0.811	0.801	0.698	0.706	0.734
3DGS-MCMC	0.919	0.925	0.928	0.930	0.932
SSS	0.921	0.926	0.929	0.932	0.934
Scene - Components	Mip-NeRF 360 - room				
Method	112k	154k	198k	242k	286k
3DGS	0.720	0.769	0.791	0.813	0.815
GES	0.786	0.797	0.824	0.851	0.839
3DHGS	0.721	0.746	0.783	0.793	0.844
3DGS-MCMC	0.911	0.916	0.920	0.923	0.925
SSS	0.912	0.918	0.923	0.926	0.928
Scene - Components	Mip-NeRF 360 - stump				
Method	32k	44.8k	57.6k	70.4k	83.2k
3DGS	0.421	0.428	0.439	0.447	0.450
GES	0.391	0.391	0.384	0.376	0.371
3DHGS	0.404	0.414	0.424	0.423	0.424
3DGS-MCMC	0.594	0.624	0.650	0.668	0.683
SSS	0.588	0.616	0.638	0.657	0.670

Table 8. **SSIM results of varying component numbers experiments for every scene in the Mip-NeRF 360 dataset.**

Scene - Components	Mip-NeRF 360 - average				
Method	δ	1.4δ	1.8δ	2.2δ	2.6δ
3DGS	0.516	0.494	0.493	0.476	0.461
GES	0.485	0.466	0.439	0.419	0.409
3DHGS	0.511	0.502	0.507	0.493	0.461
3DGS-MCMC	0.284	0.264	0.250	0.239	0.231
SSS	0.284	0.264	0.249	0.237	0.228
Scene - Components	Mip-NeRF 360 - bicycle				
Method	54k	75k	97k	118k	140k
3DGS	0.640	0.637	0.632	0.631	0.636
GES	0.640	0.642	0.648	0.646	0.642
3DHGS	0.645	0.642	0.644	0.639	0.644
3DGS-MCMC	0.430	0.405	0.387	0.371	0.360
SSS	0.434	0.412	0.394	0.379	0.367
Scene - Components	Mip-NeRF 360 - bonsai				
Method	206k	280k	360k	440k	520k
3DGS	0.388	0.360	0.361	0.352	0.340
GES	0.370	0.342	0.311	0.284	0.242
3DHGS	0.384	0.366	0.360	0.347	0.343
3DGS-MCMC	0.200	0.189	0.183	0.178	0.175
SSS	0.192	0.181	0.173	0.168	0.164
Scene - Components	Mip-NeRF 360 - counter				
Method	155k	224k	288k	352k	416k
3DGS	0.499	0.476	0.477	0.427	0.352
GES	0.461	0.359	0.284	0.266	0.242
3DHGS	0.513	0.495	0.480	0.438	0.364
3DGS-MCMC	0.212	0.200	0.192	0.187	0.183
SSS	0.214	0.197	0.188	0.181	0.176
Scene - Components	Mip-NeRF 360 - garden				
Method	138k	196k	252k	308k	364k
3DGS	0.661	0.640	0.627	0.613	0.601
GES	0.652	0.635	0.623	0.607	0.594
3DHGS	0.649	0.638	0.623	0.606	0.541
3DGS-MCMC	0.296	0.256	0.229	0.209	0.194
SSS	0.297	0.257	0.228	0.207	0.191
Scene - Components	Mip-NeRF 360 - kitchen				
Method	241k	336k	432k	528k	624k
3DGS	0.347	0.312	0.353	0.344	0.334
GES	0.235	0.260	0.211	0.163	0.156
3DHGS	0.295	0.308	0.413	0.406	0.378
3DGS-MCMC	0.145	0.134	0.129	0.125	0.122
SSS	0.142	0.132	0.125	0.119	0.115
Scene - Components	Mip-NeRF 360 - room				
Method	112k	154k	198k	242k	286k
3DGS	0.471	0.429	0.403	0.376	0.375
GES	0.404	0.396	0.361	0.325	0.342
3DHGS	0.467	0.449	0.413	0.401	0.343
3DGS-MCMC	0.235	0.223	0.215	0.209	0.203
SSS	0.230	0.217	0.207	0.200	0.195
Scene - Components	Mip-NeRF 360 - stump				
Method	32k	44.8k	57.6k	70.4k	83.2k
3DGS	0.610	0.605	0.595	0.590	0.588
GES	0.634	0.632	0.639	0.642	0.647
3DHGS	0.627	0.619	0.612	0.612	0.613
3DGS-MCMC	0.474	0.442	0.415	0.395	0.379
SSS	0.481	0.451	0.427	0.406	0.390

Table 9. **LPIPS results of varying component numbers experiments for every scene in the Mip-NeRF 360 dataset.**

process (transformation and marginalization) in Sec. 4.1.

Scene - Components	Tanks&Temples - average				
Method	δ	1.4δ	1.8δ	2.2δ	2.6δ
3DGS	15.70	15.79	15.88	16.37	16.64
GES	16.74	17.47	17.80	18.38	18.71
3DHGS	15.13	15.26	15.36	15.55	15.69
3DGS-MCMC	23.18	23.53	23.65	23.84	23.93
SSS	23.67	24.03	24.19	24.35	24.40
Scene - Components	Tanks&Temples - train				
Method	182k	252k	324k	396k	468k
3DGS	15.20	15.53	15.58	16.45	16.92
GES	18.15	19.39	19.94	21.14	21.58
3DHGS	14.42	14.62	14.75	15.05	15.24
3DGS-MCMC	21.72	22.07	22.03	22.24	22.34
SSS	22.46	22.74	22.84	22.92	22.97
Scene - Components	Tanks&Temples - truck				
Method	136k	196k	252k	308k	364k
3DGS	16.20	16.06	16.18	16.29	16.36
GES	15.32	15.54	15.66	15.62	15.85
3DHGS	15.84	15.90	15.97	16.05	16.13
3DGS-MCMC	24.64	24.99	25.27	25.44	25.53
SSS	24.88	25.33	25.53	25.74	25.88
Scene - Components	Deep Blending - average				
Method	δ	1.4δ	1.8δ	2.2δ	2.6δ
3DGS	15.95	16.05	16.33	16.60	17.06
GES	15.99	16.10	16.55	17.38	17.23
3DHGS	16.22	16.63	16.54	18.02	18.07
3DGS-MCMC	28.38	28.82	29.11	29.17	29.35
SSS	28.69	29.08	29.29	29.41	29.67
Scene - Components	Deep Blending - drjohnson				
Method	80k	112k	144k	176k	208k
3DGS	16.66	16.64	17.05	16.91	17.34
GES	16.43	16.84	17.13	17.41	18.05
3DHGS	17.17	17.29	17.37	19.19	19.82
3DGS-MCMC	28.15	28.62	28.83	28.90	28.94
SSS	28.79	29.05	29.19	29.43	29.42
Scene - Components	Deep Blending - playroom				
Method	37k	51.8k	66.6k	81.4k	96.2k
3DGS	15.23	15.45	15.61	16.28	16.78
GES	15.56	15.36	15.96	17.34	16.42
3DHGS	15.26	15.96	15.71	16.22	16.94
3DGS-MCMC	28.61	29.02	29.39	29.45	29.76
SSS	28.58	29.10	29.40	29.40	29.91

Table 10. **PSNR results of varying component numbers experiments for every scene in Tanks&Temples and Deep Blending dataset.**

Scene - Components	Tanks&Temples - average				
Method	δ	1.4δ	1.8δ	2.2δ	2.6δ
3DGS	0.528	0.537	0.543	0.566	0.580
GES	0.589	0.623	0.642	0.668	0.680
3DHGS	0.504	0.511	0.518	0.526	0.533
3DGS-MCMC	0.826	0.836	0.842	0.848	0.851
SSS	0.827	0.839	0.846	0.852	0.856
Scene - Components	Tanks&Temples - train				
Method	182k	252k	324k	396k	468k
3DGS	0.506	0.521	0.523	0.561	0.584
GES	0.645	0.699	0.727	0.774	0.788
3DHGS	0.473	0.479	0.485	0.494	0.505
3DGS-MCMC	0.798	0.807	0.814	0.820	0.824
SSS	0.799	0.812	0.821	0.828	0.833
Scene - Components	Tanks&Temples - truck				
Method	136k	196k	252k	308k	364k
3DGS	0.549	0.554	0.563	0.571	0.576
GES	0.532	0.548	0.557	0.562	0.572
3DHGS	0.535	0.543	0.550	0.557	0.561
3DGS-MCMC	0.855	0.865	0.871	0.876	0.879
SSS	0.854	0.865	0.872	0.877	0.880
Scene - Components	Deep Blending - average				
Method	δ	1.4δ	1.8δ	2.2δ	2.6δ
3DGS	0.702	0.706	0.711	0.715	0.726
GES	0.704	0.707	0.717	0.732	0.732
3DHGS	0.704	0.713	0.710	0.737	0.737
3DGS-MCMC	0.881	0.887	0.890	0.893	0.895
SSS	0.881	0.887	0.891	0.893	0.897
Scene - Components	Deep Blending - drjohnson				
Method	80k	112k	144k	176k	208k
3DGS	0.698	0.701	0.707	0.707	0.717
GES	0.696	0.705	0.715	0.721	0.735
3DHGS	0.706	0.711	0.711	0.752	0.744
3DGS-MCMC	0.880	0.885	0.888	0.890	0.891
SSS	0.880	0.885	0.890	0.894	0.896
Scene - Components	Deep Blending - playroom				
Method	37k	51.8k	66.6k	81.4k	96.2k
3DGS	0.706	0.711	0.714	0.723	0.735
GES	0.713	0.709	0.719	0.742	0.730
3DHGS	0.701	0.714	0.709	0.722	0.731
3DGS-MCMC	0.883	0.890	0.893	0.896	0.899
SSS	0.882	0.888	0.892	0.891	0.897

Table 11. **SSIM results of varying component numbers experiments for every scene in Tanks&Temples and Deep Blending dataset.**

For backward propagation with training, we derive the relevant partial derivatives with respect to Student's t distribution, which is given in Sec. 4.2. Theoretically, the value of ν of Student's t distribution can be infinite, but in practice, to avoid numerical issues, we limit the range of ν from 1 to 10000.

There are two major differences between using Student's t distribution and Gaussian distribution. The first is that a t-distribution is not a t-distribution anymore after convolution with another t-distribution. So, we did not add a low-pass filter like in vanilla 3DGS (adding 0.3 to the diagonal values of the projected covariance matrix) in our final model. However, we do add the low-pass filter as a practical solution in SGD + positive t-distribution (replacing Gaussian with Student's t but without negative components and SGHMC) in the Ablation Study for fair comparison with similar component numbers. This will be discussed in Sec. 3.2. Another difference is that Student's t distribution does not have a general "empirical rule" as Gaussian distribution. In vanilla 3DGS, they use the "empirical rule" (also known as "68–95–99.7 rule" or "three-sigma rule") to truncate the projected Gaussian in 2D space. The "three-

Scene - Components	Tanks&Temples - average				
Method	δ	1.4δ	1.8δ	2.2δ	2.6δ
3DGS	0.569	0.558	0.552	0.527	0.510
GES	0.500	0.460	0.441	0.410	0.397
3DHGS	0.590	0.582	0.574	0.566	0.558
3DGS-MCMC	0.231	0.214	0.203	0.195	0.187
SSS	0.229	0.210	0.195	0.185	0.177
Scene - Components	Tanks&Temples - train				
Method	182k	252k	324k	396k	468k
3DGS	0.564	0.548	0.543	0.504	0.476
GES	0.415	0.352	0.321	0.267	0.251
3DHGS	0.597	0.592	0.583	0.575	0.561
3DGS-MCMC	0.255	0.240	0.229	0.220	0.212
SSS	0.254	0.235	0.220	0.208	0.199
Scene - Components	Tanks&Temples - truck				
Method	136k	196k	252k	308k	364k
3DGS	0.575	0.569	0.560	0.551	0.545
GES	0.585	0.569	0.561	0.552	0.543
3DHGS	0.583	0.571	0.564	0.558	0.555
3DGS-MCMC	0.206	0.189	0.178	0.170	0.163
SSS	0.204	0.184	0.170	0.161	0.154
Scene - Components	Deep Blending - average				
Method	δ	1.4δ	1.8δ	2.2δ	2.6δ
3DGS	0.525	0.522	0.515	0.510	0.500
GES	0.522	0.522	0.509	0.495	0.491
3DHGS	0.522	0.515	0.515	0.490	0.488
3DGS-MCMC	0.325	0.310	0.301	0.295	0.290
SSS	0.319	0.305	0.296	0.289	0.282
Scene - Components	Deep Blending - drjohnson				
Method	80k	112k	144k	176k	208k
3DGS	0.526	0.523	0.514	0.512	0.504
GES	0.529	0.522	0.508	0.503	0.487
3DHGS	0.515	0.515	0.511	0.476	0.482
3DGS-MCMC	0.316	0.290	0.290	0.284	0.280
SSS	0.310	0.297	0.288	0.280	0.275
Scene - Components	Deep Blending - playroom				
Method	37k	51.8k	66.6k	81.4k	96.2k
3DGS	0.524	0.521	0.515	0.508	0.496
GES	0.516	0.522	0.509	0.488	0.494
3DHGS	0.530	0.515	0.519	0.505	0.494
3DGS-MCMC	0.333	0.320	0.312	0.306	0.301
SSS	0.329	0.314	0.304	0.298	0.289

Table 12. **LPIPS results of varying component numbers experiments for every scene in Tanks&Temples and Deep Blending dataset.**

sigma rule" can be applied to t-distributions with thin tails (high ν degree). Besides, we obtain different critical values of different ν degrees [5] and interpolate these to obtain appropriate truncated values for fat-tailed t-distributions with lower ν values.

Our training process does not use the adaptive density control in vanilla 3DGS. Instead, we recycle components with low opacity to high opacity components every n iterations. This is done by creating a Multinomial distribution of opacity values and sampling all components that have relatively high probability. Recycle can only be achieved if certain conditions are met, *i.e.* the change to the current state (rendering result) after the recycle is minimal [8, 12].

Ablation Setup — Metric	PSNR ↑	SSIM↑	LPIPS↓
Mip-NeRF	22.22	0.759	0.257
3DGS	23.14	0.841	0.183
GES	23.35	0.836	0.198
SGD + positive t-dis	23.80	0.838	0.191
SGHMC + positive t-dis	24.53	0.864	0.155
Full model	24.87	0.873	0.138

Table 13. **Ablation Study on Tanks&Temples with the same component numbers as in baselines. More details are in the SM.**

Method \ Scene	bicycle	bonsai	counter	garden	kitchen	room	stump	average
SGD + positive t-dis	25.52	31.80	28.82	27.60	30.93	31.60	26.72	29.00
SGHMC + positive t-dis	25.97	32.87	29.49	27.92	31.92	32.04	27.44	29.66
Full SSS model	25.68	33.50	29.87	28.09	32.43	32.57	27.17	29.90

Table 14. **PSNR results of ablation study for every scene in Mip-NeRF 360 dataset.**

Method \ Scene	bicycle	bonsai	counter	garden	kitchen	room	stump	average
SGD + positive t-dis	0.767	0.941	0.909	0.866	0.927	0.923	0.771	0.872
SGHMC + positive t-dis	0.801	0.952	0.920	0.879	0.936	0.933	0.817	0.891
Full SSS model	0.798	0.956	0.926	0.882	0.939	0.938	0.813	0.893

Table 15. **SSIM results of ablation study for every scene in Mip-NeRF 360 dataset.**

Method \ Scene	bicycle	bonsai	counter	garden	kitchen	room	stump	average
SGD + positive t-dis	0.225	0.185	0.188	0.110	0.121	0.199	0.234	0.180
SGHMC + positive t-dis	0.182	0.159	0.168	0.099	0.112	0.181	0.183	0.155
Full SSS model	0.173	0.151	0.156	0.009	0.104	0.167	0.174	0.145

Table 16. **LPIPS results of ablation study for every scene in Mip-NeRF 360 dataset.**

Dataset - Scene	Tanks&Temples			Deep Blending		
Method	train	truck	average	drjohnson	playroom	average
SGD + positive t-dis	22.18	25.42	23.80	29.22	29.93	29.57
SGHMC + positive t-dis	22.92	26.15	24.53	29.45	30.04	29.75
Full SSS model	23.32	26.41	24.87	29.66	30.47	30.07

Table 17. **PSNR results of ablation study for every scene in Tanks&Temples and Deep Blending dataset.**

Dataset - Scene	Tanks&Temples			Deep Blending		
Method	train	truck	average	drjohnson	playroom	average
SGD + positive t-dis	0.803	0.874	0.838	0.900	0.901	0.901
SGHMC + positive t-dis	0.838	0.891	0.864	0.902	0.905	0.903
Full SSS model	0.850	0.897	0.873	0.905	0.909	0.907

Table 18. **SSIM results of ablation study for every scene in Tanks&Temples and Deep Blending dataset.**

Dataset - Scene	Tanks&Temples			Deep Blending		
Method	train	truck	average	drjohnson	playroom	average
SGD + positive t-dis	0.226	0.156	0.191	0.248	0.247	0.247
SGHMC + positive t-dis	0.186	0.124	0.155	0.262	0.257	0.260
Full SSS model	0.166	0.109	0.138	0.249	0.245	0.247

Table 19. **LPIPS results of ablation study for every scene in Tanks&Temples and Deep Blending dataset.**

Dataset	Tanks&Temples			Deep Blending		
Method—Metric	PSNR ↑	SSIM↑	LPIPS↓	PSNR ↑	SSIM↑	LPIPS↓
3DGS	23.14	0.841	0.183	29.41	0.903	0.243
SGD + positive t-dis	23.80	0.838	0.191	29.57	0.901	0.247
SGHMC + 3DGS	24.52	0.869	0.150	29.56	0.906	0.239
SGHMC + positive t-dis	24.53	0.864	0.155	29.75	0.903	0.259
Full model	24.87	0.873	0.138	30.07	0.907	0.247

Table 20. **More Ablation Study.**

Dataset	Mip-NeRF360		Tanks&Temples		Deep Blending	
Method—Metric	FPS	Training time	FPS	Training time	FPS	Training time
3DGS	99	21min	120	12min	119	21min
3DGS-MCMC	79	32min	105	17min	138	29min
SSS	71	45min	87	33min	100	21min

Table 21. **Training time and rendering efficiency.**

We give detailed formulae of recycling in Sec. 5. The result of calculating the new covariance matrix after recycling includes the $\beta()$ function. Because CUDA native implementation does not include the support for the $\beta()$ function, we decompose the $\beta()$ function into $\Gamma()$ function for computation. Further, to prevent excessive values by the $\Gamma()$ function with a large input number, we further use the $\ln(\Gamma())$ function in practice. These are shown with Eqs. (34) and (35).

Finally, we employ the Adam gradient in SGHMC. The hyperparameters in our training process are largely the same as those of the original 3DGS and 3DGS-MCMC (*e.g.* we down-scale images to the same resolution for large scenes in the Mip-NeRF360 dataset as most works did), but because we use Student's t, negative components, and SGHMC, there are some new parameters. Detailed parameter values can be found in our code. **The code is available at https://github.com/realcrane/3D-student-splating-and-scooping.**

All our experiments are running with one NVIDIA RTX 4090 GPU. We show training/rendering time in Tab. 21. SSS is slower compared to vanilla 3DGS but still achieves real-time rendering (>70fps).

3.1. Adaptation of Baselines

To illustrate the parameter efficiency of our model, we did experiments with fewer components amount all baselines and our model. Because other baselines except 3DGS-MCMC do not support setting the maximum number of components, we modified their code. These modifications are minimal and do not involve any adjustment of hyperparameters to ensure a fair comparison. Specifically, we modified the adaptive density control used by most baselines. We stop densification (cloning and splitting) if the current number of components has reached the maximum setting. Note that due to the existence of the pruning strategy, the number of components can also be decreased, which will allow densification to continue until the preset densification iteration is reached.

3.2. Ablation Study Implementation

We show our Ablation study results in Sec. 1.3. These include three settings with one or more components in SSS to evaluate their effectiveness. Setting 1 (SGD + positive t-dis) uses adaptive density control and SGD optimization in vanilla 3DGS with positive Student's t distribution. Setting 2 (SGHMC + positive t-dis) uses SGHMC only with positive Student's t distribution. Setting 3 (full SSS model) uses SGHMC with both positive and negative Student's t distribution. Since Student's t distribution does not have an analytical form after convolution with another Student's t, we remove the low-pass filter in the splatting algorithm [17, 18]. The main function of the low-pass filter is to ensure that the minimum scale of each component is close to 1 pixel. Removing the low-pass filter does not affect our final result, because our principled sampler can ensure that the final components are of the appropriate size. However, for setting 1, the absence of a low-pass filter will lead to an increase in the number of components (considering that it will use more tiny components to reconstruct more details). In order to ensure the fairness of the comparison (to make our number of components roughly equal to that of vanilla 3DGS), we added a small value (0.3) to the covariance matrix of the t-distribution after projection as a low-pass filter from an engineering perspective. Furthermore, we do not present the results of using adaptive density control and SGD optimization in vanilla 3DGS with both positive and negative t-distributions. This is because our negative components are designed to be used with our principled sampler. The densification of positive components in vanilla 3DGS is based on the size of the components and their gradients, which is not applicable to negative components. Adding negative components directly to adaptive density control will lead to worse results and is not the focus of our study. Finally, we also test our SGHMC sampler with Gaussian distributions to show its advantage. We use the same hyperparameters related to SGHMC and keep all other hyperparameters the same as vanilla 3DGS in this experiment.

4. Forward and Backward Passes

Although the general forward/backward passes of SSS are similar to 3DGS, the equations are different. This is mainly due to the introduction of t-distribution. Below we give details of mathematical derivation.

4.1. Forward Pass

Affine transformation of 3D Student's t distribution A 3D Student's t distribution is:

$$T(x;\mu,\Sigma,\nu) = \frac{\Gamma(\frac{\nu+3}{2})}{(\nu\pi)^{\frac{3}{2}}\Gamma(\frac{\nu}{2})|\Sigma|^{\frac{1}{2}}} \cdot [1+\frac{1}{\nu}(x-\mu)^T\Sigma^{-1}(x-\mu)]^{-\frac{\nu+3}{2}} \tag{1}$$

where $\nu \geq 1$, $x, \mu \in \mathbb{R}^3$ $\Sigma \in \mathbb{R}^{3\times3}$ are the control parameter, the mean and the covariance matrix, which describes the spread and orientation of the distribution in 3D space. To render an image, Eq. (1) needs to be projected into a 2D image plane, which goes through a view transformation W, d, and a (approximate) projective transformation J [17, 18], so a 3D point x after transformation becomes u:

$$\begin{aligned} u &= m^{-1}(x) = A^{-1}(x-b) \\ \text{where } A &= JW \text{ and } b = x + J(d-t) \end{aligned} \tag{2}$$

where t is the camera coordinates. Applying the transformations Eq. (2) to Eq. (1) gives:

$$\begin{aligned} T(x) &= \frac{\Gamma(\frac{\nu+3}{2})}{(\nu\pi)^{\frac{3}{2}}\Gamma(\frac{\nu}{2})|\Sigma|^{\frac{1}{2}}} \\ &\cdot [1+\frac{1}{\nu}(A^{-1}(x-b)-\mu)^T\Sigma^{-1}(A^{-1}(x-b)-\mu)]^{-\frac{\nu+3}{2}} \\ &= \frac{\Gamma(\frac{\nu+3}{2})}{(\nu\pi)^{\frac{3}{2}}\Gamma(\frac{\nu}{2})|\Sigma|^{\frac{1}{2}}} \\ &\cdot [1+\frac{1}{\nu}(A^{-1}x-A^{-1}b-A^{-1}A\mu)^T \\ &\cdot \Sigma^{-1}(A^{-1}x-A^{-1}b-A^{-1}A\mu)]^{-\frac{\nu+3}{2}} \\ &= \frac{\Gamma(\frac{\nu+3}{2})}{(\nu\pi)^{\frac{3}{2}}\Gamma(\frac{\nu}{2})|\Sigma|^{\frac{1}{2}}} \\ &\cdot [1+\frac{1}{\nu}(x-b-A\mu)^T(A^{-1})^T\Sigma^{-1}A^{-1}(x-b-A\mu)]^{-\frac{\nu+3}{2}} \\ &= \frac{\Gamma(\frac{\nu+3}{2})}{(\nu\pi)^{\frac{3}{2}}\Gamma(\frac{\nu}{2})|\Sigma|^{\frac{1}{2}}} \\ &\cdot [1+\frac{1}{\nu}(x-m(\mu))^T(A\Sigma A^T)^{-1}(x-m(\mu))]^{-\frac{\nu+3}{2}} \\ &= \frac{\Gamma(\frac{\nu+3}{2})|A|}{(\nu\pi)^{\frac{3}{2}}\Gamma(\frac{\nu}{2})|\Sigma|^{\frac{1}{2}}|A|^{\frac{1}{2}}|A^T|^{\frac{1}{2}}} \\ &\cdot [1+\frac{1}{\nu}(x-m(\mu))^T(A\Sigma A^T)^{-1}(x-m(\mu))]^{-\frac{\nu+3}{2}} \\ &= |A|\frac{\Gamma(\frac{\nu+3}{2})}{(\nu\pi)^{\frac{3}{2}}\Gamma(\frac{\nu}{2})|A\Sigma A^T|^{\frac{1}{2}}} \\ &\cdot [1+\frac{1}{\nu}(x-m(\mu))^T(A\Sigma A^T)^{-1}(x-m(\mu))]^{-\frac{\nu+3}{2}} \\ &= |A|T(x,m(\mu),A\Sigma A^T,\nu) \end{aligned} \tag{3}$$

So for $r(x) = T(x;\mu,\Sigma,\nu)$ at x, its transformed density is:

$$\begin{aligned} r_k^{'}(x) &= \frac{1}{|A^{-1}|}T(x;m(\mu),A\Sigma A^T,\nu) \\ &= \frac{1}{|W^{-1}J_k^{-1}|}T(x;m(\mu),JW\Sigma W^TJ^T,\nu) \\ &= \frac{1}{|W^{-1}J^{-1}|}T(x;m(\mu),\Sigma^{'},\nu) \end{aligned} \tag{4}$$

where $\Sigma^{'} = JW\Sigma W^TJ^T$.

This result still holds after we drop the normalization constant $\frac{\Gamma(\frac{\nu+3}{2})}{(\nu\pi)^{\frac{3}{2}}\Gamma(\frac{\nu}{2})|\Sigma|^{\frac{1}{2}}}$.

Integral along a ray (marginalization) After deriving the view and projective transformation for Student's t distribution, we need to integrate it along the ray that intersects with it for rendering. *i.e.* splatting it. This is actually the marginalization of Student's t distribution along one dimension, and it can be done by another affine transformation $y = Wx$, where $W = \begin{bmatrix}\mathbf{I},0\\0,0\end{bmatrix}$ with $\mathbf{I}$ is a 2×2 identify matrix. To see this, imagine x can be divided into two parts $x = \begin{bmatrix}x_1\\x_2\end{bmatrix}$, so that:

$$T(Wx;\mu,\Sigma,\nu,p) = T(y;W\mu,W\Sigma W^T,\nu,p) \tag{5}$$

and if $\mu = \begin{bmatrix}\mu_a\\\mu_b\end{bmatrix}$ and $\Sigma = \begin{bmatrix}\Sigma_{aa} & \Sigma_{ab}\\\Sigma_{ba} & \Sigma_{bb}\end{bmatrix}$ then

$$T(x;\mu,\Sigma,\nu,p) = T(\begin{pmatrix}x_a\\x_b\end{pmatrix};\begin{pmatrix}\mu_a\\\mu_b\end{pmatrix},\begin{pmatrix}\Sigma_{aa} & \Sigma_{ab}\\\Sigma_{ba} & \Sigma_{bb}\end{pmatrix},\nu,p) \tag{6}$$

where means by transforming x with $W = \begin{bmatrix}\mathbf{I},0\\0,0\end{bmatrix}$, we will get the marginal distribution of x_a:

$$\begin{aligned} T(Wx;\mu,\Sigma,\nu,p) &= T(y;W\mu,W\Sigma W^T,\nu,p) \\ &= T(x_a;\mu_a,\Sigma_{aa},\nu,p_a) \end{aligned} \tag{7}$$

This way we can marginalize any variable for T. Integration along a ray gives a 2D t-distribution:

$$T_{2D}(x,\mu,\Sigma,\nu) = [1+\frac{1}{\nu}(x-\mu)^T\Sigma^{-1}(x-\mu)]^{-\frac{\nu+2}{2}} \tag{8}$$

where x now is in 2D space.

So far, we have derived all key equations for the forward pass when using t-distributions as the mixture components.

4.2. Backward Pass

The most important computation in the backward pass is to compute all the key gradients, $\frac{\partial L}{\partial \mu}$, $\frac{\partial L}{\partial S}$,$\frac{\partial L}{\partial R}$, $\frac{\partial L}{\partial c}$, $\frac{\partial L}{\partial o}$, and

$\frac{\partial L}{\partial \nu}$, where L is the loss, μ is the mean of a t-distribution. S and R are the scaling and rotation matrices for the covariance matrix of t-distribution. c is the color represented in spherical harmonics. o is the opacity. ν is the control parameter of the t-distribution.

L is calculated between the ground-truth pixel colors and rendered colors. The rendered pixel colors are:

$$c(x) = \sum_{i=1}^{N} c_i o_i T_i^{2D}(x) \prod_{j=1}^{i-1}(1 - o_j T_j^{2D}(x)) \tag{9}$$

where x is the position of a pixel in 2D images.

For simplicity, we let $\alpha = oT$. Based on the chain rule:

$$\frac{\partial L}{\partial \mu} = \frac{\partial L}{\partial_{rgb}} \cdot \frac{\partial_{rgb}}{\partial_\alpha} \cdot \frac{\partial_\alpha}{\partial_{T^{2D}}} \cdot \frac{\partial_{T^{2D}}}{\partial_h} \cdot \frac{\partial_h}{\partial_{\mu'}} \cdot \frac{\partial_{\mu'}}{\partial_\mu} \tag{10}$$

$$\frac{\partial_{Loss}}{\partial_o} = \frac{\partial_{Loss}}{\partial_{rgb}} \cdot \frac{\partial_{rgb}}{\partial_\alpha} \cdot \frac{\partial_\alpha}{\partial_o} \tag{11}$$

$$\frac{\partial L}{\partial_c} = \frac{\partial L}{\partial_{rgb}} \cdot \frac{\partial_{rgb}}{\partial_c} \tag{12}$$

$$\frac{\partial L}{\partial_S} = \frac{\partial L}{\partial_{rgb}} \cdot \frac{\partial_{rgb}}{\partial} \cdot \frac{\partial_\alpha}{\partial_{T^{2D}}} \cdot \frac{\partial_{T^{2D}}}{\partial_h} \cdot \frac{\partial_h}{\partial_{\Sigma'}} \cdot \frac{\partial_{\Sigma'}}{\partial_\Sigma} \cdot \frac{\partial_\Sigma}{\partial_S} \tag{13}$$

$$\frac{\partial L}{\partial_R} = \frac{\partial L}{\partial_{rgb}} \cdot \frac{\partial_{rgb}}{\partial_\alpha} \cdot \frac{\partial_\alpha}{\partial_{T^{2D}}} \cdot \frac{\partial_{T^{2D}}}{\partial_h} \cdot \frac{\partial_h}{\partial_{\Sigma'}} \cdot \frac{\partial_{\Sigma'}}{\partial_\Sigma} \cdot \frac{\partial_\Sigma}{\partial_R} \tag{14}$$

$$\frac{\partial L}{\partial_\nu} = \frac{\partial L}{\partial_{rgb}} \cdot \frac{\partial_{rgb}}{\partial_\alpha} \cdot \frac{\partial_\alpha}{\partial_{T^{2D}}} \cdot \frac{\partial_{T^{2D}}}{\partial_\nu} \tag{15}$$

where μ' and Σ' are the projected μ and Σ in 2D space. h is a predefined function explained shortly.

For these gradients, we only need to replace the calculation of Gaussian in vanilla 3DGS with the calculation of Student's t distribution based on the chain rule, except for the gradient for ν in Eq. (15). In order to further simplify the calculation, we extract the same part of the 2D Gaussian and 2D Student's t distributions, and define a function $h(x)$:

$$\begin{aligned} G^{2D}(x, \mu', \Sigma') &= exp^{-\frac{1}{2}h(x)} \\ T^{2D}(x, \mu', \Sigma', \nu) &= [1 + \frac{1}{\nu}h(x)]^{-\frac{\nu+2}{2}} \end{aligned} \tag{16}$$

where

$$h(x) = (x - \mu')^T \Sigma'^{-1}(x - \mu') \tag{17}$$

We have

$$\begin{aligned} \frac{\partial T^{2D}(x, \mu', \Sigma', \nu)}{\partial h(x)} &= [1 + \frac{1}{\nu}h(x)]^{-\frac{\nu+2}{2}} \quad dh(x) \\ &= -\frac{\nu+2}{2} \cdot \frac{1}{\nu} \cdot [1 + \frac{1}{\nu}h(x)]^{-\frac{\nu+4}{2}} \quad dh(x) \end{aligned} \tag{18}$$

to replace every

$$\frac{\partial G^{2D}(x, \mu', \Sigma')}{\partial h(x)} = -\frac{1}{2} exp^{-\frac{1}{2}h(x)} \quad dh(x) \tag{19}$$

Then we can use the derived calculations in 3DGS for the rest of the parts, we refer the readers to read [15] for more detailed mathematics if interested.

Finally, the gradient for optimizing ν needs to be calculated separately. Assuming $g(\nu) = 1 + \frac{1}{\nu}h(x)$, we have

$$\begin{aligned} \frac{\partial_{T^{2D}(x,\mu',\Sigma',\nu)}}{\partial_\nu} &= [1 + \frac{1}{\nu}(x - \mu')^T \Sigma'^{-1}(x - \mu')]^{-\frac{\nu+2}{2}} d\nu \\ &= [1 + \frac{1}{\nu}h(x)]^{-\frac{\nu+2}{2}} d\nu \\ &= g(\nu)^{-\frac{\nu+2}{2}} d\nu \\ &= -\frac{\nu+2}{2} g(\nu)^{-\frac{\nu+4}{2}} \cdot g^{'}(\nu) \\ &= -\frac{\nu+2}{2}(1 + \frac{1}{\nu}h(x))^{-\frac{\nu+4}{2}} \\ &\cdot ((1 + \frac{1}{\nu}h(x))d\nu) \\ &= -\frac{\nu+2}{2}(1 + \frac{1}{\nu}h(x))^{-\frac{\nu+4}{2}} \cdot (-\frac{h(x)}{\nu^2}) \\ &= \frac{\nu+2}{2} \cdot \frac{h(x)}{\nu^2} \left[1 + \frac{1}{\nu}h(x)\right]^{-\frac{\nu+4}{2}} \\ &= \frac{\nu+2}{2} \cdot \frac{(x - \mu')^T \Sigma'^{-1}(x - \mu')}{\nu^2} \\ &\cdot \left[1 + \frac{1}{\nu}(x - \mu')^T \Sigma'^{-1}(x - \mu')\right]^{-\frac{\nu+4}{2}} \end{aligned} \tag{20}$$

5. Component Recycling

The key equation for relocating low opacity components to a high opacity component is to make sure the distribution before and after the relocation is not changed [8, 12]. If we move new components to the location of an old component $\mu_{new} = \mu_{old}$, this is ensured by separately handling the opacity and the covariance matrix. For opacity, it is simply:

$$(1 - O_{new})^N = (1 - O_{old}) \tag{21}$$

For covariance:

$$\begin{aligned} &minimize \int_{-\infty}^{\infty} ||C_{new}(x) - C_{old}(x)||dx \text{ or} \\ &minimize|| \int_{-\infty}^{\infty} C_{new}(x) - \int_{-\infty}^{\infty} C_{old}(x)||dx \end{aligned} \tag{22}$$

To solve Eq. (22), we first need to separately derive $\int_{-\infty}^{\infty} C_{old}(x)$ and $\int_{-\infty}^{\infty} C_{new}(x)$. For $\int_{-\infty}^{\infty} C_{old}(x)$, assuming $u = \frac{x}{\sqrt{\nu_{old}\Sigma_{old}}}$, $du = \frac{1}{\sqrt{\nu_{old}\Sigma_{old}}}dx$, and $dx =$

$\sqrt{\nu_{old}\Sigma_{old}}du$, then:

$$\begin{aligned}\int_{-\infty}^{\infty} C_{old}(x) &= \int_{-\infty}^{\infty} o_{old}[1+\frac{1}{\nu_{old}}\frac{x^2}{\Sigma_{old}}]^{-\frac{\nu_{old}+3}{2}}dx \\ &= o_{old}\int_{-\infty}^{\infty}[1+\frac{1}{\nu_{old}}\frac{x^2}{\Sigma_{old}}]^{-\frac{\nu_{old}+3}{2}}dx \\ &= o_{old}\int_{-\infty}^{\infty}[1+u^2]^{-\frac{\nu_{old}+3}{2}}\sqrt{\nu_{old}\Sigma_{old}}du \\ &= o_{old}\sqrt{\nu_{old}\Sigma_{old}}\int_{-\infty}^{\infty}[1+u^2]^{-\frac{\nu_{old}+3}{2}}du\end{aligned} \tag{23}$$

For the form $\int_{-\infty}^{\infty}[1+x^2]^{-\alpha}dx$, assuming $x = tan(\theta)$, $dx = sec^2(\theta)d\theta$, and $1+x^2 = 1+tan^2(\theta) = sec^2(\theta)$, then:

$$\begin{aligned}&\int_{-\infty}^{\infty}[1+x^2]^{-\alpha}dx \\ &= 2\int_0^{\infty}[1+x^2]^{-\alpha}dx \\ &= 2\int_0^{\pi/2}(sec^2(\theta))^{-\alpha}sec^2(\theta)d\theta \\ &= 2\int_0^{\pi/2}(sec^{2-2\alpha}(\theta))d\theta \\ &= 2\int_0^{\pi/2}(cos^{2\alpha-2}(\theta))d\theta\end{aligned} \tag{24}$$

Further, according to the definition of the β function:

$$\beta(x,y) = 2\int_0^{\pi/2} sin^{2x-1}(\theta)cos^{2y-1}(\theta)d\theta \tag{25}$$

Eq. (24) becomes:

$$\begin{aligned}&2\int_0^{\pi/2}(cos^{2\alpha-2}(\theta))d\theta \\ &2\int_0^{\pi/2} sin^{2\frac{1}{2}-1}(\theta)cos^{2(\alpha-\frac{1}{2})-1}(\theta)d\theta \\ &= \beta(\frac{1}{2},(\alpha-\frac{1}{2}))\end{aligned} \tag{26}$$

Then Eq. (23) becomes:

$$\begin{aligned}&o_{old}\sqrt{\nu_{old}\Sigma_{old}}\int_{-\infty}^{\infty}[1+u^2]^{-\frac{\nu_{old}+3}{2}}du \\ &= o_{old}\sqrt{\nu_{old}\Sigma_{old}}\cdot\beta(\frac{1}{2},\frac{\nu_{old}+2}{2})\end{aligned} \tag{27}$$

For $\int_{-\infty}^{\infty} C_{new}(x)$,

$$\begin{aligned}\int_{-\infty}^{\infty} C_{new}(x) &= \int_{-\infty}^{\infty}\sum_{i=1}^{N} o_{new}[1+\frac{1}{\nu_{new}}\frac{x^2}{\Sigma_{new}}]^{-\frac{\nu_{new}+3}{2}} \\ &\cdot(1-o_{new}[1+\frac{1}{\nu_{new}}\frac{x^2}{\Sigma_{new}}]^{-\frac{\nu_{new}+3}{2}})^{i-1}dx\end{aligned} \tag{28}$$

From Binomial theorem:

$$(x+y)^n = \sum_{k=0}^{n}\binom{n}{k}x^{n-k}y^k = \sum_{k=0}^{n}\binom{n}{k}x^k y^{n-k} \tag{29}$$

Eq. (28) becomes:

$$\begin{aligned}&\int_{-\infty}^{\infty}\sum_{i=1}^{N} o_{new}[1+\frac{1}{\nu_{new}}\frac{x^2}{\Sigma_{new}}]^{-\frac{\nu_{new}+3}{2}} \\ &\cdot(1-o_{new}[1+\frac{1}{\nu_{new}}\frac{x^2}{\Sigma_{new}}]^{-\frac{\nu_{new}+3}{2}})^{i-1}dx \\ &= \int_{-\infty}^{\infty}\sum_{i=1}^{N} o_{new}[1+\frac{1}{\nu_{new}}\frac{x^2}{\Sigma_{new}}]^{-\frac{\nu_{new}+3}{2}} \\ &\cdot\sum_{k=0}^{i-1}\binom{i-1}{k}(-o_{new}[1+\frac{1}{\nu_{new}}\frac{x^2}{\Sigma_{new}}]^{-\frac{\nu_{new}+3}{2}})^k dx \\ &= \int_{-\infty}^{\infty}\sum_{i=1}^{N} o_{new}[1+\frac{1}{\nu_{new}}\frac{x^2}{\Sigma_{new}}]^{-\frac{\nu_{new}+3}{2}} \\ &\cdot\sum_{k=0}^{i-1}\binom{i-1}{k}(-1)^k(o_{new})^k([1+\frac{1}{\nu_{new}}\frac{x^2}{\Sigma_{new}}]^{-\frac{\nu_{new}+3}{2}})^k dx \\ &= \int_{-\infty}^{\infty}\sum_{i=1}^{N}\sum_{k=0}^{i-1}\binom{i-1}{k}(-1)^k(o_{new})^{k+1} \\ &\cdot([1+\frac{1}{\nu_{new}}\frac{x^2}{\Sigma_{new}}]^{-\frac{(k+1)(\nu_{new}+3)}{2}})dx \\ &= \sum_{i=1}^{N}\sum_{k=0}^{i-1}\binom{i-1}{k}(-1)^k(o_{new})^{k+1} \\ &\cdot\int_{-\infty}^{\infty}([1+\frac{1}{\nu_{new}}\frac{x^2}{\Sigma_{new}}]^{-\frac{(k+1)(\nu_{new}+3)}{2}})dx\end{aligned} \tag{30}$$

According to equations 24, 25 and 26, Eq. (30) becomes:

$$\begin{aligned}&\sum_{i=1}^{N}\sum_{k=0}^{i-1}\binom{i-1}{k}(-1)^k(o_{new})^{k+1} \\ &\cdot\int_{-\infty}^{\infty}([1+\frac{1}{\nu_{new}}\frac{x^2}{\Sigma_{new}}]^{-\frac{(k+1)(\nu_{new}+3)}{2}})dx \\ &= \sum_{i=1}^{N}\sum_{k=0}^{i-1}\binom{i-1}{k}(-1)^k(o_{new})^{k+1} \\ &\cdot\sqrt{\nu_{new}\Sigma_{new}}\cdot\beta(\frac{1}{2},\frac{(k+1)(\nu_{new}+3)-1}{2})\end{aligned} \tag{31}$$

Having derived $\int_{-\infty}^{\infty} C_{old}(x)$ and $\int_{-\infty}^{\infty} C_{new}(x)$, we mini-

mize Eq. (22) by setting:

$$
\begin{aligned}
&\int_{-\infty}^{\infty} C_{new}(x) = \int_{-\infty}^{\infty} C_{old}(x) \\
\Rightarrow &\sum_{i=1}^{N}\sum_{k=0}^{i-1}\binom{i-1}{k}(-1)^k(o_{new})^{k+1} \\
&\cdot \sqrt{\nu_{new}\Sigma_{new}}\beta(\frac{1}{2}, \frac{(k+1)(\nu_{new}+3)-1}{2}) \\
&= o_{old}\sqrt{\nu_{old}\Sigma_{old}}\beta(\frac{1}{2}, \frac{\nu_{old}+2}{2}) \\
\Rightarrow &\sqrt{\nu_{new}\Sigma_{new}}\sum_{i=1}^{N}\sum_{k=0}^{i-1}\binom{i-1}{k}(-1)^k(o_{new})^{k+1} \\
&\cdot \beta(\frac{1}{2}, \frac{(k+1)(\nu_{new}+3)-1}{2}) \\
&= o_{old}\sqrt{\nu_{old}\Sigma_{old}}\beta(\frac{1}{2}, \frac{\nu_{old}+2}{2}) \\
\Rightarrow &\Sigma_{new} = (o_{old})^2\frac{\nu_{old}}{\nu_{new}} \\
&(\frac{\beta(\frac{1}{2}, \frac{\nu_{old}+2}{2})}{\sum_{i=1}^{N}\sum_{k=0}^{i-1}\binom{i-1}{k}(-1)^k(o_{new})^{k+1}\beta(\frac{1}{2}, \frac{(k+1)(\nu_{new}+3)-1}{2})})^2 \\
&\cdot \Sigma_{old}
\end{aligned}
\tag{32}
$$

So at the end, we can compute Σ_{new} based on Σ_{old}:

$$
\begin{aligned}
&\Sigma_{new} = (o_{old})^2\frac{\nu_{old}}{\nu_{new}} \\
&(\frac{\beta(\frac{1}{2}, \frac{\nu_{old}+2}{2})}{\sum_{i=1}^{N}\sum_{k=0}^{i-1}\binom{i-1}{k}(-1)^k(o_{new})^{k+1}\beta(\frac{1}{2}, \frac{(k+1)(\nu_{new}+3)-1}{2})})^2 \\
&\cdot \Sigma_{old}
\end{aligned}
\tag{33}
$$

Furthermore, since β function can be represented by Γ functions:

$$
\beta(x, y) = \frac{\Gamma(x)\Gamma(y)}{\Gamma(x+y)} \tag{34}
$$

We can use the Γ function or $\ln(\Gamma)$ function instead of β function in practice:

$$
\begin{aligned}
&\beta(x, y) = \frac{\Gamma(x)\Gamma(y)}{\Gamma(x+y)} \\
\Rightarrow &\ln(\beta(x, y)) = \ln(\Gamma(x)) + \ln(\Gamma(y)) - \ln(\Gamma(x+y)) \\
\Rightarrow &\beta(x, y) = exp(\ln(\Gamma(x)) + \ln(\Gamma(y)) - \ln(\Gamma(x+y)))
\end{aligned}
\tag{35}
$$

6. SGHMC Sampling

In SGHMC [3], the posterior distribution of model parameters θ given a set of independent observations $x \in D$ is defined as

$$
\pi(\theta, r) \propto exp(-U(\theta)) \tag{36}
$$

where $U(\theta)$ is a potential energy function which is $-\sum_{x\in D}\log p(x|\theta) - \log p(\theta)$.

To sample from $p(\theta|D)$, The Hamiltonian (Hybrid) Monte Carlo (HMC) considers generating samples from a joint distribution of $\pi(\theta, r)$ defined by

$$
\pi(\theta, r) \propto exp(-U(\theta) - \frac{1}{2}r^T M r) \tag{37}
$$

where the Hamiltonian function is defined by $H(\theta, r) = U(\theta) + \frac{1}{2}r^T M r$. M is the mass matrix and r is the auxiliary momentum variables. Further, to introduce stochastic gradients into the sampling, the Stochastic Gradient Hamiltonian Monte Carlo (SGHMC) is proposed in [3] where an additional friction is introduced. We refer the readers to [3] for detailed mathematical derivation.

To employ SGHMC for our SSS sampling, we start by parameterizing a joint distribution:

$$
P(\theta, r) \propto exp(-L_\theta(x) - \frac{1}{2}r^T I r) \tag{38}
$$

By defining a similar Hamiltonian function as in Eq. (37) and adding a friction term as in [3], we derive

$$
\begin{aligned}
d\theta =& M^{-1}rdt \\
dr =& -\nabla U(\theta)dt - CM^{-1}rdt + \mathcal{N}(0, 2Cdt)
\end{aligned}
\tag{39}
$$

Next, we further modify the updating equations in Eq. (39) to:

$$
\begin{aligned}
\mu_{t+1} &= \mu_t - \varepsilon^2\left[\frac{\partial L}{\partial \mu}\right]_t + F + N \\
F &= \sigma(o)\varepsilon(1-\varepsilon C)r_{t-1} \\
N &= \sigma(o)\mathcal{N}(0, 2\varepsilon^{\frac{3}{2}}C) \\
r_{t+1} &= r_t - \varepsilon\left[\frac{\partial L}{\partial \mu}\right]_{t+1} - \varepsilon C r_{t-1} + \mathcal{N}(0, 2\varepsilon C) \\
&\text{where } \sigma(o) = \sigma(-k(o-t))
\end{aligned}
\tag{40}
$$

where ε is the learning rate and decays during learning. $\mathcal{N}$ is Gaussian noise. o is the opacity. To further clarify the relation between the learnable parameter and the momentum, we first show the updating rule for the learnable parameter in the original SGHMC:

$$
\begin{aligned}
\mu_{t+1} &= \mu_t + \varepsilon * (r_t - \varepsilon * G_t - \varepsilon * C * r_t + N(0, 2 * \varepsilon * C)) \\
&= \mu_t + \varepsilon * r_t - \varepsilon^2 * G_t - \varepsilon^2 * C * r_t + N(0, 2 * \varepsilon^{\frac{3}{2}} * C) \\
&= \mu_t - \varepsilon^2 * G_t + \varepsilon * (1 - \varepsilon * C) * r_t + N(0, 2 * \varepsilon^{\frac{3}{2}} * C)
\end{aligned}
\tag{41}
$$

where G is the gradient $\left[\frac{\partial L}{\partial \mu}\right]$.

Before Stochastic Gradient Hamiltonian Monte Carlo (SGHMC), we first attempted the Stochastic Gradient Langevin Dynamics (SGLD) sampling in 3DGS-MCMC [8]. Although it outperforms the standard optimization employed in the original 3DGS and its variants, it still

sometimes generates suboptimal results. Other than the randomness in the optimization itself, we suspected the core reason is the increased model complexity, especially the introduction of ν in t-distribution, which brings tight coupling between many parameters, *e.g.* ν greatly influencing μ and Σ. On the high level, the optimization of μ and Σ can be seen as seeking the optimal distribution within a family of distributions. In 3DGS, this family is Gaussians. However, when ν is also optimized, the family itself changes during optimization. This is the core reason we resort to SGHMC which shows better sampling behaviors given tightly coupled parameters [3].

Furthermore, we leave the learning of ν and other parameters to Adam, as this can help further decouple the parameters. This is a similar strategy to the 3DGS-MCMC. This is complemented by using SGHMC on the location of t-distribution μ. Also, for components with high opacity, we tend to think that they are near their local optima, so no further random perturbation is needed. This is achieved by adding a sigmoid switch:

$$\mu_{t+1} = \mu_t - \varepsilon^2 * G_t + \sigma(\varepsilon * (1 - \varepsilon * C) * r_t) + \sigma(N(0, 2 * \varepsilon^{\frac{3}{2}} * C)) \tag{42}$$

σ is the customized sigmoid function. Note we add the sigmoid switch to both the friction and the noise, partially to keep the integrity of the sampler and partially to remove the friction for nearly optimal components. Also, when the friction and noise are removed, the parameter is updated by $\varepsilon^2 * G_t$, *i.e.* the gradient scaled by ε^2 which is much smaller than the learning rate ε, encouraging local search.

Finally, we conducted experiments between our SGHMC and SGLD in 3DGS-MCMC. We found that while SGLD can explore large spaces, SGHMC is better at local exploitation. To achieve the best results, we finally performed a burn-in stage with the friction removed during training for large exploration. In order to maintain the anisotropy of Σ of Student's t distribution after the friction is removed, we multiply the noise by Σ following 3DGS-MCMC. After the burn-in stage, we add the friction back and restore the noise (no longer multiplied by Σ).

7. Representation Limitation

Although we demonstrate the strength of our approach in both qualitative and quantitative evaluations, we do acknowledge that our approach is not perfect in every scenario. We have discussed the limitations of SSS in the main context. For example, the Student's t distribution is limited by symmetric and smooth representation, which makes it difficult to handle sharp shapes perfectly. Student's t distribution combined with negative components can increase the representation ability, but the range of representation is still limited. In addition, although the randomness of SGHMC brings more exploration of space, it still sometimes suffer from the floating artifact problem commonly observed in 3DGS.

References

[1] Jonathan T Barron, Ben Mildenhall, Matthew Tancik, Peter Hedman, Ricardo Martin-Brualla, and Pratul P Srinivasan. Mip-nerf: A multiscale representation for anti-aliasing neural radiance fields. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 5855–5864, 2021. 1

[2] Jonathan T Barron, Ben Mildenhall, Dor Verbin, Pratul P Srinivasan, and Peter Hedman. Mip-nerf 360: Unbounded anti-aliased neural radiance fields. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 5470–5479, 2022. 1, 2

[3] Tianqi Chen, Emily Fox, and Carlos Guestrin. Stochastic gradient hamiltonian monte carlo. In *International conference on machine learning*, pages 1683–1691. PMLR, 2014. 13, 14

[4] Abdullah Hamdi, Luke Melas-Kyriazi, Jinjie Mai, Guocheng Qian, Ruoshi Liu, Carl Vondrick, Bernard Ghanem, and Andrea Vedaldi. Ges: Generalized exponential splatting for efficient radiance field rendering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 19812–19822, 2024. 1, 3

[5] N Alan Heckert, James J Filliben, C M Croarkin, B Hembree, William F Guthrie, P Tobias, and J Prinz. Handbook 151: Nist/sematech e-handbook of statistical methods. 2002. 8

[6] Peter Hedman, Julien Philip, True Price, Jan-Michael Frahm, George Drettakis, and Gabriel Brostow. Deep blending for free-viewpoint image-based rendering. *ACM Transactions on Graphics (ToG)*, 37(6):1–15, 2018. 1, 2

[7] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 3d gaussian splatting for real-time radiance field rendering. *ACM Trans. Graph.*, 42(4):139–1, 2023. 1, 3, 5

[8] Shakiba Kheradmand, Daniel Rebain, Gopal Sharma, Weiwei Sun, Yang-Che Tseng, Hossam Isack, Abhishek Kar, Andrea Tagliasacchi, and Kwang Moo Yi. 3d gaussian splatting as markov chain monte carlo. *Advances in Neural Information Processing Systems*, 37:80965–80986, 2024. 1, 3, 5, 8, 11, 13

[9] Arno Knapitsch, Jaesik Park, Qian-Yi Zhou, and Vladlen Koltun. Tanks and temples: Benchmarking large-scale scene reconstruction. *ACM Transactions on Graphics (ToG)*, 36 (4):1–13, 2017. 1, 2

[10] Haolin Li, Jinyang Liu, Mario Sznaier, and Octavia Camps. 3d-hgs: 3d half-gaussian splatting. *arXiv preprint arXiv:2406.02720*, 2024. 1, 3

[11] Tao Lu, Mulin Yu, Linning Xu, Yuanbo Xiangli, Limin Wang, Dahua Lin, and Bo Dai. Scaffold-gs: Structured 3d gaussians for view-adaptive rendering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 20654–20664, 2024. 1

[12] Samuel Rota Bulò, Lorenzo Porzi, and Peter Kontschieder. Revising densification in gaussian splatting. In *European*

Conference on Computer Vision, pages 347–362. Springer, 2024. 8, 11

[13] Shimon Ullman. The interpretation of structure from motion. *Proceedings of the Royal Society of London. Series B. Biological Sciences*, 203(1153):405–426, 1979. 3

[14] Zhou Wang, Alan C Bovik, Hamid R Sheikh, and Eero P Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE transactions on image processing*, 13(4):600–612, 2004. 1

[15] Vickie Ye and Angjoo Kanazawa. Mathematical supplement for the gsplat library. *arXiv preprint arXiv:2312.02121*, 2023. 11

[16] Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 586–595, 2018. 1

[17] Matthias Zwicker, Hanspeter Pfister, Jeroen Van Baar, and Markus Gross. Ewa volume splatting. In *Proceedings Visualization, 2001. VIS'01.*, pages 29–538. IEEE, 2001. 5, 9, 10

[18] Matthias Zwicker, Hanspeter Pfister, Jeroen Van Baar, and Markus Gross. Ewa splatting. *IEEE Transactions on Visualization and Computer Graphics*, 8(3):223–238, 2002. 5, 9, 10