3D Student Splatting and Scooping

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

Abstract

Recently, 3D Gaussian Splatting (3DGS) provides a new framework for novel view synthesis, and has spiked a new wave of research in neural rendering and related applications. As 3DGS is becoming a foundational component of many models, any improvement on 3DGS itself can bring huge benefits. To this end, we aim to improve the fundamental paradigm and formulation of 3DGS. We argue that as an unnormalized mixture model, it needs to be neither Gaussians nor splatting. We subsequently propose a new mixture model consisting of flexible Student's t distributions, with both positive (splatting) and negative (scooping) densities. We name our model Student Splatting and Scooping, or SSS. When providing better expressivity, SSS also poses new challenges in learning. Therefore, we also propose a new principled sampling approach for optimization. Through exhaustive evaluation and comparison, across multiple datasets, settings, and metrics, we demonstrate that SSS outperforms existing methods in terms of quality and parameter efficiency, e.g. *achieving matching or better quality with similar numbers of components, and obtaining comparable results while reducing the component number by as much as 82%.*

1. Introduction

Presented initially as a neural rendering technique, 3D Gaussian Splatting (3DGS) [13] has quickly become a versatile component in various systems, *e.g.* geometry reconstruction, autonomous driving [3, 6]. Given its importance as a foundational component, very recently researchers start to investigate possible alternatives to the basic framework of 3DGS, *e.g.* more expressive distributions instead of Gaussians [8, 18], more principled optimization [14], which all focus on improving the model expressivity. Our research is among these attempts.

The key to 3DGS' success lies in its name: Gaussian and splatting. 3DGS can be seen as a (unnormalized) Gaussian mixture, which provides two advantages. As a general-purpose distribution, Gaussians can approximate an arbitrary density function hence good expressivity. Also, Gaussians have analytical forms under *e.g.* affine transformation, enabling quick evaluation in 3D-2D projection, thus quick learning from images. Meanwhile, splatting provides a flexible way of identifying only the relevant Gaussians to an image for learning. Despite the success, the framework can still suffer from insufficient expressivity [16, 34], and low parameter efficiency, *i.e.* needing a large number of components [17, 28]. Therefore, we re-examine the three key components in 3DGS: Gaussian, splatting, and the optimization. Since its underlying principle is essentially to fit a 3D mixture model to a radiance field, we argue it needs not to be restricted to Gaussians or splatting.

To this end, we propose a simple yet effective generalization of 3DGS. We first replace Gaussians with Student's t distribution with one degree of freedom, referred to simply as t-distribution. Similar to Gaussians, t-distribution also enjoys good properties such as analytical forms under affine transformation. More importantly, t-distribution can be seen as a generalization of Gaussians and therefore is more expressive. t-distribution has a control parameter for the tail fatness, representing distributions ranging from Cauchy distribution to Gaussian distribution, and any distribution inbetween. Compared with Gaussians, Cauchy is fat-tailed, *i.e.* a single Cauchy can cover a larger area with comparatively higher densities than a Gaussian. Furthermore, by making this control parameter learnable, we learn components with a wide range of varying tail thicknesses.

Next, we extend the splatting scheme which only operates in the positive density space. Inspired by mixture models with negative components [20], we propose to employ both positive and negative components to splat (adding) and scoop (subtracting) densities from the model. This leads to more complex mathematical forms than 3DGS but we derive their close-form gradients for learning.

Finally, as the increased model complexity, optimization methods based on naive stochastic gradient descent become insufficient, due to parameter coupling. Therefore, we propose a principled sampling scheme based on Stochastic

*Corresponding author, he_wang@ucl.ac.uk

Gradient Hamiltonian Monte Carlo (SGHMC).

We refer to our model as Student Splatting and Scooping (SSS). SSS is evaluated on multiple datasets and compared with existing methods. Experiments show that SSS achieves higher quality often with fewer number of components, demonstrating more expressivity and higher parameter efficiency. Formally, our contributions include:

- A new model named Student Splatting and Scooping (SSS), which is highly expressive and parameter efficient.
- A new mixture model with flexible components learned from a set of distribution families for neural rendering.
- A mixture model with negative components, which extends the learning into the negative density space.
- A principled sampling approach to tackle parameter coupling during learning.

2. Related Work

3D Reconstruction and Novel View Synthesis 3D reconstruction and Novel View Synthesis have been long-standing research topics in computer vision. Traditional methods mainly include Multi-View Stereo (MVS) [27] and Structure from Motion (SFM) [30]. Recently, the advent of Deep Learning has brought important changes to the field. In particular, the techniques based on Neural Radiance Field (NeRF) [22] and 3D Gaussian Splatting (3DGS) [13] have set new state-of-the-art (SOTA) benchmarks.

NeRF methods NeRF [22] proposes to implicitly encode the radiance field of a 3D object/scene into a neural network and renders the 3D geometry and textures through a continuous volume rendering function. Since then, a large number of methods based on NeRF have been proposed, namely NeRF++ [38], Mip-NeRF [2] and Mip-NeRF360 [2] to improve rendering quality, Plenoxels [7] and Instant-NGP [23] to accelerate NeRF training, D-NeRF [25] to extend NeRFs to dynamic scenes, DreamFusion [24] and Zero-1-to-3 [19] to employ it for text-to-3d generation models, etc. However, the biggest drawback of NeRF is that the ray casting process for rendering is time consuming. Despite the effort in improving its rendering efficiency *e.g.* SNeRG [10] and mobileNeRF [5], it still cannot be used for real-time rendering in most cases.

Splatting methods 3DGS [13] solves the above problem for real-time rendering, by replacing the volume rendering with a differentiable rasterization method, which achieves the SOTA render quality. 3DGS uses 3D Gaussian as the primitive for the splatting method [40, 41]. It directly projects 3D Gaussians onto the 2D image plane through view/projective transformation for rasterization. Similar to NeRFs, prolific follow up research has been conducted based on 3DGS. GS++ [12] and Mip-Splatting [36] aim to improve rendering quality, 4D Gaussian Splatting [33] and Deformable 3D Gaussians [35] extend 3DGS to dynamic scenes, Dreamgaussian [29] employs 3DGS for text-to-3D tasks. One particular line of research is to improve the fundamental paradigm of 3DGS. This includes FreGS [37], 3DGS-MCMC [14] and Bulò et al. [26] which optimize the training process and adaptive density control in 3DGS, Scaffold-GS [21] and Implicit Gaussian Splatting [34] which combine grid representation with 3DGS for better rendering quality. More recently, there is also research exploring different primitives other than 3D Gaussians. 2DGS [11] obtains better surface reconstruction by changing the primitives from 3D Gaussian to 2D Gaussian for aligning the 3D scene. GES [8] uses a generalized exponential kernel to increase the expression ability of primitives and reduce memory cost. 3DHGS [18] decomposes one Gaussian into two half-Gaussians to obtain asymmetry and better expressivity.

Our research is among the few recent efforts in improving the fundamental formulation of 3DGS. Different from them, we propose to use more expressive and flexible distributions, 3D Student's t distribution, as the basic primitive. In addition, we also use both positive and negative densities to extend the optimization into the negative density space for better representation. Finally, we propose a principled sampling approach for learning, deviating from most of the above research.

3. Methodology

3.1. Preliminaries: 3DGS as a mixture model

3DGS essentially fits a (unnormalized) 3D Gaussian mixture model to a radiance field [13]:

$$P(x) = \sum w_i G_i(x),\ G(x) = e^{-\frac{1}{2}(x-\mu)^T \Sigma^{-1}(x-\mu)} \quad (1)$$

where $w_i > 0$. μ is the center position of Gaussian. $\Sigma \in \mathbb{R}^{3\times3}$ is the covariance matrix, parameterized by a scaling matrix S and a rotation matrix R to maintain its positive semi-definiteness: $\Sigma = RSS^TR^T$. Additionally, every 3D Gaussian is associated with opacity $o \in [0, 1]$ and color $c \in \mathbb{R}^{27}$ which is represented by spherical harmonics and view-dependent [13].w_i is determined by o, c and compositing values after projection.

Since the mixture can only be evaluated in 2D, when rendering an image, a 3D Gaussian is projected onto the 2D image plane, G^{2D}, via integrating it in the camera view space along a ray, for computing a pixel color:

$$C(u) = \sum_{i=1}^{N} c_i o_i G_i^{2D}(u) \prod_{j=1}^{i-1} (1 - o_j G_j^{2D}(u)). \quad (2)$$

where N is the number of the Gaussians that intersect with the ray cast from the pixel u. Finally, the Gaussian parameters, opacity, and colors are learned based on the observed

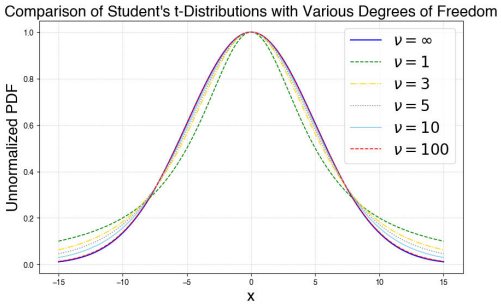


Figure 1. Student's t with varying degrees of freedom ν. (standard deviation is 5).

2D images. Eq. (2) reveals that $C(u)$ can be seen as a 2D Gaussian mixture, except that now a component weight is also a function of other components, introducing additional cross-component interactions.

Numerically, Gaussians, as the mixture component, are closed under affine transformation and marginalization of variables, so that the forward/backward pass can be quickly computed. 3DGS is a *monotonic* mixture as it is additive, *i.e.* $w_i > 0$. Due to the success of 3DGS, existing works have since followed this paradigm [12, 21, 36, 37].

3.2. Student's t as a basic component

We propose an unnormalized t-distribution mixture model, where a t-distribution is defined by a mean (location) $\mu \in \mathbb{R}^3$, a covariance matrix (shape) $\Sigma = RSS^TR^T \in \mathbb{R}^{3\times3}$, a degree of freedom (tail-fatness) $\nu \in [1, +\infty)$, associated with opacity o, and color c:

$$P(x) = \sum w_i T_i(x)\ ,\ w_i > 0$$
$$T(x|\nu) = [1 + \frac{1}{\nu}(x-\mu)^T\Sigma^{-1}(x-\mu)]^{-\frac{\nu+3}{2}}, \quad (3)$$

where we can drop the scalar $\frac{\Gamma(\nu+3)/2}{\Gamma(\nu/2)\nu^{\frac{3}{2}}\pi^{\frac{3}{2}}|\Sigma|^{\frac{1}{2}}}$ in the original t-distribution safely to facilitate learning. The choice is driven by two main factors. First, t-distribution is a strong generalization of Gaussians. As shown in Fig. 1, when $\nu \to 1, T \to Cauchy$; when $\nu \to \infty, T \to Gaussian$. So t-distribution can capture what Gaussians capture and beyond. Furthermore, since Cauchy is fat-tailed, it can cover larger areas with higher densities than Gaussians therefore potentially reducing the number of components. As ν, μ and Σ are learnable, SSS becomes a mixture of components learned from an infinite number of distribution families, instead of one family [13], providing further flexibility.

The second reason for t-distribution is it also provides good properties similar to Gaussians, *e.g.* close under affine transformation and marginalization of variables. Rendering a pixel requires an affine transformation, then a projective transformation, followed by an integration along a ray, to be applied to a component, which has a simple form in 3DGS. In SSS, t-distribution also has a close form:

$$T^{2D}(u) = [1 + \frac{1}{\nu}(u-\mu^{2D})^T(\Sigma^{2D})^{-1}(u-\mu^{2D})]^{-\frac{\nu+2}{2}}$$
$$\mu^{2D} = (W\mu + t)_{1:2}/((W\mu+t)_3)$$
$$\Sigma^{2D} = (JW\Sigma W^TJ^T)_{1:2,1:2}, \quad (4)$$

where the subscripts select the corresponding rows and columns. W, t and J are the affine transformation (*i.e.* scale, translation) and (approximated) projective transformation [40, 41]. This enables us to easily derive the key gradients for learning shown in the supplementary material (SM), unlike existing research also using alternative mixture components but requiring approximation [8].

In summary, the mixture of learnable t-distributions enhances the representational power and provides good mathematical properties for learning.

3.3. Splatting and Scooping

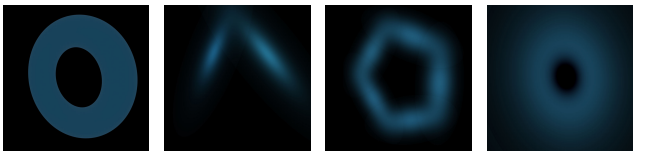
While monotonic mixture models are powerful, a *non-monotonic* mixture model recently has been proposed by introducing negative components [20], arguing that it is sub-optimal to only operate in the positive density space:

$$P(x) = (\sum w_i T_i(x))^2 = \sum_{i=1}^{K}\sum_{j=1}^{K} w_i w_j T_i(x) T_j(x) \quad (5)$$

where $w \in \mathbb{R}$. In our problem, a negative density makes good sense as it can be seen as subtracting a color. However, our experiments using Eq. (5) show that it is not ideal as it introduces interactions between every pair of components, increasing the model evaluation complexity to $O(n^2)$ where n is the number of components in the model, making it significantly slower than before. Therefore, we still use Eq. (3) but with $w \in \mathbb{R}$ instead of $w > 0$ where $w_i = c_i o_i \prod_{k=1}^{i-1}(1 - o_k T_k^{2D}(u))$ and $o \in [-1, 1]$. Normally this might cause issues as Eq. (3) is then not well defined with negative components. However, we can parameterize the density in an energy-based form explained later which is well defined. In learning, we constrain the opacity by a $tanh$ function so that positive and negative components can dynamically change signs while being bounded. Introducing negative t-distribution can enhance the representation power and the parameter efficiency. We show a simple experiment in Fig. 2, where fewer components are needed to fit the shape topology of a torus. In SSS, a component with negative densities is equivalent to removing its color from the mixture. Negative components are particularly useful in subtracting colors.

3.4. Learning via sampling

Recently, it is argued that principled sampling is better in 3DGS, *e.g.* Markov Chain Monte Carlo [14], instead of



(a) Topology data (b) 2 positive components (c) 5 positive components (d) 1 positive and 1 negative components

Figure 2. **High parameter efficiency by negative components**. We use a torus with only ambient lighting and frontal views (a), where the challenge is to capture the shape topology with as few components as possible. We initialize the component means near the center. Only using positive densities either underfits if two components are used (b), or requires at least 5 components to capture the topology correctly (c). In contrast, in (d), we only need two components (one positive and one negative), to capture the topology of the shape. Both components are co-located at the center of the torus. The positive component covers the torus but also the hole, while the negative component subtracts densities in the middle to make a hole.

naive stochastic gradient descent (SGD). Empirically, we found training SSS involves learning more tightly coupled parameters compared with 3DGS, namely among ν, μ, and Σ. We speculate that this is because changing ν in learning is changing the family of distributions within which we optimize μ and Σ. Therefore, we propose a sampling scheme that mitigates such coupling, based on Stochastic Gradient Hamiltonian Monte Carlo (SGHMC).

Starting from the Hamiltonian Monte Carlo, we first parameterize the posterior distribution as:

$$P(\theta, r) \propto exp(-L_\theta(x) - \frac{1}{2} r^T I r) \tag{6}$$

where $L_\theta(x)$ is our loss function, I is an identity matrix, r is a momentum auxiliary variable, and θ is the learnable parameters. This is because Eq. (3) with $w \in \mathbb{R}$ is not a well-defined distribution, which makes direct sampling difficult. Using an energy function circumvents this issue and prescribes the high density regions of good θ. Intuitively, we would like to sample θ to minimize $L_\theta(x)$. In addition, to decouple parameters during learning, the momentum term $\frac{1}{2} r^T I r$ creates frictions for each dimension of the parameter space, enabling adaptive learning for each parameter.

For $L_\theta(x)$, our rendering function computes the pixel value based on the N components associated with a ray:

$$C(u) = \sum_{i=1}^{N} c_i o_i T_i^{2D}(u) \prod_{j=1}^{i-1} (1 - o_j T_j^{2D}(u)). \tag{7}$$

where u is the pixel. c and o are the color and opacity associated with a component T. We then employ the following loss function [14]:

$$\begin{aligned} L =& (1 - \epsilon_{D-SSIM}) L_1 + \epsilon_{D-SSIM} L_{D-SSIM} \\ &+ \epsilon_o \sum_i |o_i|_1 + \epsilon_\Sigma \sum_i \sum_j |\sqrt{\lambda_{i,j}}|_1 \end{aligned} \tag{8}$$

where the L_1 norm, and the structural similarity L_{D-SSIM} loss aim to reconstruct images, while the last two terms act as regularization, with λ being the eigenvalues of Σ. The regularization applied to the opacity ensures that the opacity is big only when a component is absolutely needed. The regularization on λ ensures the model uses components as spiky as possible (i.e. small variances). Together, they minimize the needed number of components [14].

Furthermore, directly sampling from Eq. (6) requires the full gradient of $U = L_\theta(x) - \frac{1}{2} r^T I r$ which is not possible given the large number of training samples. Therefore, replacing the full gradient with stochastic gradient will introduce a noise term: $\nabla \hat{U} = \nabla U + \mathcal{N}(0, V)$, where $\mathcal{N}$ is Normal and V is the covariance of the stochastic gradient noise. Under mild assumptions [4], sampling Eq. (6) using stochastic gradients becomes (with detailed derivation in the SM):

$$\begin{aligned} d\theta =& M^{-1} r dt \\ dr =& -\nabla U(\theta) dt - C M^{-1} r dt + \mathcal{N}(0, 2C dt) \end{aligned} \tag{9}$$

where $\mathcal{N}$ is Gaussian noise, M is a mass matrix, and C is a control parameter dictating the friction term $CM^{-1}rdt$ and noise $\mathcal{N}(0, 2Cdt)$. In our problem, it is crucial to design good friction and noise scheduling. The effect of this principled sampling method is further discussed in SM.

3.4.1. Friction and Noise Scheduling

We first use SGHMC on μ and Adam on the other parameters. To learn μ, we modify Eq. (9) to:

$$\begin{aligned} \mu_{t+1} &= \mu_t - \varepsilon^2 \left[\frac{\partial L}{\partial \mu}\right]_t + F + N \\ F &= \sigma(o) \varepsilon (1 - \varepsilon C) r_{t-1} \\ N &= \sigma(o) \mathcal{N}(0, 2\varepsilon^{\frac{3}{2}} C) \\ r_{t+1} &= r_t - \varepsilon \left[\frac{\partial L}{\partial \mu}\right]_{t+1} - \varepsilon C r_{t-1} + \mathcal{N}(0, 2\varepsilon C) \\ &\text{where } \sigma(o) = \sigma(-k(o - t)) \end{aligned} \tag{10}$$

where ε is the learning rate and decays during learning. $\mathcal{N}$ is Gaussian noise. o is the opacity. The main difference between Eq. (10) and Eq. (9) is we now have adaptive friction F and noise N for μ. σ (sigmoid function) switches on/off the friction and noise. We use $k = 100$ and $t = 0.995$, so that it only activates for components with opacity lower than 0.005. When it is activated, friction and noise are added to these components. Note that if F is disabled, Eq. (10)

is simplified to a Stochastic Gradient Langevin Dynamics scheme [32].

When learning, we initialize with a sparse set of (SfM) points without normals [13], run Eq. (10) without F for burn-in for exploration, then run the full sampling for exploitation until convergence. During burn-in stage, we multiply N by the covariance Σ of the component to maintain the anisotropy profile of the t-distribution. After the burn-in, Σ is removed, and the anisotropy is then maintained by F due to the momentum r.

Key gradients Overall, the key learnable parameters for each component in SSS include the mean μ, covariance Σ (*i.e.* S and R), color c, opacity o, and degree parameter ν. To compute Eq. (9), the key gradients are $\frac{\partial L}{\partial \mu}$, $\frac{\partial L}{\partial S}$,$\frac{\partial L}{\partial R}$, $\frac{\partial L}{\partial c}$, $\frac{\partial L}{\partial o}$, and $\frac{\partial L}{\partial \nu}$. For simplicity, we give them in the SM.

3.4.2. Adding and Recycling Components

Components can become nearly transparent during sampling, *i.e.* near zero opacity. In 3DGS, they are removed. Recently, it is argued that they should be recycled [14], by relocating them to a high opacity component. However, careful consideration needs to be taken as the overall distribution before and after relocation should be the same [26]. When moving some components to the location of another component, this is ensured by:

$$\min \int_{-\infty}^{\infty} ||C_{new}(\mu) - C_{old}(\mu)||_2^2 d\mu \tag{11}$$

where C_{new} and C_{old} are the color after and before relocation respectively. Minimizing this integral ensures the smallest possible pixel-wise color changes over the whole domain. Minimizing Eq. (11) in SSS leads to:

$$\begin{aligned}
&\mu_{new} = \mu_{old}\ , (1 - o_{new})^N = (1 - o_{old}) \\
&\Sigma_{new} = (o_{old})^2 \frac{\nu_{old}}{\nu_{new}} (\frac{\beta(\frac{1}{2}, \frac{\nu_{old}+2}{2})}{K})^2 \Sigma_{old} \\
&K = \sum_{i=1}^{N} \sum_{k=0}^{i-1} \binom{i-1}{k} (-1)^k (o_{new})^{k+1} Z) \\
&Z = \beta(\frac{1}{2}, \frac{(k+1)(\nu_{new}+3)-1}{2})
\end{aligned} \tag{12}$$

μ_{new} and μ_{old}, o_{new} and o_{old}, Σ_{new} and Σ_{old} are the mean, opacity, and covariance after and before relocation respectively. N is the total number of components after relocation, *i.e.* moving N-1 low opacity components to the location of 1 high opacity component. β is the beta function. We leave the detailed derivation in the SM. Note we do not distinguish between positive and negative components during relocation. This introduces a perturbation on the sign of the opacity. In Eq. (12), if $o_{old} > 0$ then all $o_{new} > 0$, or otherwise $o_{new} < 0$ if $o_{old} < 0$, regardless their original opacity signs. This sign perturbation in practice helps the mixing of the sampling. Furthermore, to ensure the sampling stability, we limit the relocation to a maximum of 5% of the total components at a time. Finally, we also add new components when needed, but do not use the adaptive density control (clone and split) in 3DGS [13]. Instead, we add 5% new components with zero opacity and then recycle them.

4. Experiments

4.1. Experimental setting

Datasets and Metrics Following existing research, we employ 11 scenes from 3 datasets, including 7 public scenes from Mip-NeRF 360 [2], 2 outdoor scenes from Tanks & Temples [15], and 2 indoor scenes from Deep Blending [9]. Also, following the previous evaluation metrics, we use Peak Signal-to-Noise Ratio (PSNR), Structural Similarity Index Metric (SSIM) [31], and Learned Perceptual Image Patch Similarity (LPIPS) [39]. We provide average scores of each dataset, and detailed scores are in the SM.

Baselines Due to there being many publications based on 3DGS, we only choose the original 3DGS [13] and the most recent work that focuses on improving the fundamental paradigm of 3DGS and has achieved the best performances. The methods include Generalized Exponential Splatting (GES) [8] and 3D Half-Gaussian Splatting (3DHGS) [18] which also use different (positive only) mixture components, Scaffold-GS [21] and Fre-GS [37] which optimize the training procedure to achieve faster convergence and better results, 3DGS-MCMC [14] which proposes a principled MCMC sampling process, and Mip-NeRF [1] which is a state-of-the-art method with Neural Radiance Field (NeRF) [22]. Overall, our baselines comprehensively include methods with new mixture components, new optimization approaches, and the SOTA quality.

The results of all baselines in general benchmarking are from their papers. In addition, we run their codes with other settings for more comparison. Since not all baseline methods are implemented in exactly the same setting, we need to adapt them for comparison. These details are in the SM.

4.2. General Benchmarks

We first compare SSS with the baselines on all scenes in their default settings, shown in Tab. 1. SSS achieves overall the best results on 6 of the 9 metrics, and the second best on 2 metrics. The only exception is the LPIPS in Deep Blending, where the difference between SSS and the best is 7×10^{-3}. Furthermore, when investigating individual scenes, SSS achieves the largest leading margin on Train. It achieves 23.23/0.844/0.170, where the second best method 3DHGS achieves 22.95/0.827/0.197, in PSNR, SSIM, and LPIPS, which is a 1.22%/2.05%/13.7% improvement. De-

Dataset	Mip-NeRF360 Dataset			Tanks&Temples			Deep Blending		
Method—Metric	PSNR ↑	SSIM↑	LPIPS↓	PSNR ↑	SSIM↑	LPIPS↓	PSNR ↑	SSIM↑	LPIPS↓
Mip-NeRF	29.23	0.844	0.207	22.22	0.759	0.257	29.40	0.901	0.245
3DGS	28.69	0.870	0.182	23.14	0.841	0.183	29.41	0.903	0.243
GES	26.91	0.794	0.250	23.35	0.836	0.198	29.68	0.901	0.252
3DHGS	29.56	0.873	0.178	24.49	0.857	0.169	29.76	0.905	0.242
Fre-GS	27.85	0.826	0.209	23.96	0.841	0.183	29.93	0.904	0.240
Scaffold-GS	28.84	0.848	0.220	23.96	0.853	0.177	30.21	0.906	0.254
3DGS-MCMC	29.89	0.900	0.190	24.29	0.860	0.190	29.67	0.890	0.320
Ours	29.90	0.893	0.145	24.87	0.873	0.138	30.07	0.907	0.247

Table 1. **Comparison.** 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.

tailed scores for each scene are in the SM. We show qualitative comparison in Figure 3.

4.3. Parameter Efficiency

SSS has stronger representation power than 3DGS and its variants. The varying tail-thickness of its components enables SSS to fit the data with fewer components, *i.e*. higher parameter efficiency. We show this via experiments under different component numbers.

Since the SfM initialization gives different components in different scenes and a method normally increases the component number during learning, we introduce a coefficient to describe the latter as a multiplicity of the former. Denoting the initial component number as δ, we test δ, 1.4δ, 1.8δ, 2.2δ, and 2.6δ as the maximum component number for comparison. Note even with 2.6δ, the component number is still much smaller than the experiments in Tab. 1. Specifically, the 2.6δ vs the original 3DGS component number are 468k/1.1m, 364k/2.6m, 208k/3.4m, 96k/2.5m, 140k/5.9m, 520k/1.3m, 416k/1.2m, 364k/5.2m, 624k/1.8m, 286k/1.5m, 83k/4.75m, in Train, Truck, DrJohnson, Playroom, Bicycle, Bonsai, Counter, Garden, Kitchen, Room, Stump, corresponding to merely 42.5%, 14%, 6.1%, 3.8%, 2.4%, 40%, 34.7%, 7%, 34.7%, 19.1%, 1.7% of the original components, a maximum of 98.3% reduction.

PSNR is averaged over the scenes in each dataset and shown in Figure 4. First, SSS outperforms all other methods in 15 out of 15 settings, demonstrating strong expressivity across all scenarios. Furthermore, when the component number decreases, all methods deteriorate, but SSS deteriorates slowly comparatively, demonstrating that SSS can fit the data much more efficiently than the rest. One specific example is the Tanks & Temples in Tab. 1. SSS achieves 23.6 PSNR with merely 180k components, already surpassing Mip-NeFR, 3DGS and GES. With around 300k components, SSS achieves 24.4 PSNR, which is only slightly worse than 3DHGS and 3DGS-MCMC, by a margin at the scale of 10^{-2}. Note this is a comparison with the methods in Tab. 1 where they use at least 1m components, *e.g*. 3DGS employs around 1.1m and 2.6m in Train and Truck, while SSS employs only around 364k and 468k, a maximum reduction of 82% of the components.

4.4. Qualitative Comparison

We further show a qualitative comparison in one scene across the different component numbers in Fig. 5. The ground-truth is one view from the Train. When restricting the component number to around 180k, the original 3DGS and one of the state-of-the-art methods 3DHGS show significant blur. This is likely to be caused by the struggle between stretching Gaussians to cover large areas and narrowing them to reconstruct details, given a limited number of components at proposal. Due to the optimization relying on stochastic gradient descent, a local minimum is sought where the distribution of the Gaussians is sub-optimal. Intuitively, the issue can be mitigated by more flexible components and/or better optimization. As expected, this is shown in GES and 3DGS-MCMC, where the former employs a more flexible component (generalized exponential function) while the latter improves the optimization itself by MCMC. The improvements by both methods suggest that these are the correct directions to improve the paradigm of 3DGS.

Next, SSS outperforms GES and 3DGS-MCMC visually when using 180k components. One example is the sky and the hill in the background in the left half of the image. GES creates a blurry background mixing the sky and the hill, with no discernible details, suggesting it uses a few components that are stretched to cover large areas. In contrast, 3DGS-MCMC can separate the sky from the hill. But it creates random white patches in the sky, which do not exist in the ground truth. This suggests that 3DGS-MCMC employs a relatively larger number of slim Gaussians to fit the details but meanwhile introduces additional noises. SSS not only successfully separates the sky and the hill, but simultaneously retains the homogeneous color in the sky and reconstructs the details on the hill, *e.g*. the trees and lawns.

(a) Bonsai from Mip-NeRF 360 [2] (b) Gardern from Mip-NeRF 360 [2] (c) Kitchen from Mip-NeRF 360 [2] (d) Truck from Tanks & Temples [15] (e) Playroom from Deep Blending [9]

Figure 3. **Visual comparison** Zoom-in for better visualization. (a) SSS restores better the indentations of the box lid; (b) SSS is the best at detailing windows in the upper center; (c) Only the image rendered by SSS contains the green track detail in the upper right corner; (d) SSS is the best at restoring the reflection in the front window of the truck; (e) SSS perfectly restores the light switch next to the stairs.

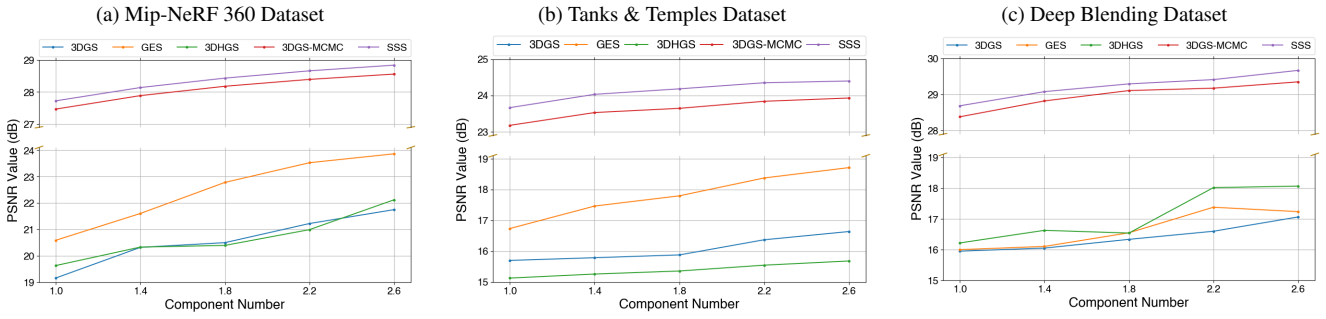


Figure 4. All methods with reduced component numbers.

This is attributed to SSS' capability of learning components with varying tail-fatness to adaptively capture large homogeneous areas and small heterogeneous regions.

Furthermore, when increasing the component number to 468k, all results are improved, as expected. 3DGS and 3DHGS still cannot work as well as other methods, suggesting they need more components. In addition, the difference between GES, 3DGS-MCMC, and SSS starts to nar-

(a) 180k components (b) 252k components (c) 324k components (d) 396k components (e) 468k components

Figure 5. **Visual results of all methods with varying component numbers.** In addition to the well-reconstructed main body of the train compared to other baselines, our method can use a small number of components to restore more details, such as plants on distant mountains, rocks on the ground nearby, etc. Besides, our sky has fewer noises and appears more similarly to the ground truth. Zoom in for details. Note that ours with 252k components has already achieved SOTA quality and beats most baselines.

row. GES can separate the sky and the hill. Both GES and 3DGS-MCMC have fewer artifacts. However, as the component number increases, there are still noticeable noises introduced to the sky, which suggests that it is a common issue for both methods when using more components to cover an area with mixed homogeneous and heterogeneous regions. Comparatively, SSS gives consistent performance across all component numbers, *i.e*. clearly separating and reconstructing the homogeneous sky and the heterogeneous hill. Also, the visual quality from 180k to 468k does not change significantly for SSS, but is noticeably improved for GES and 3DGS-MCMC, suggesting a higher parameter efficiency of SSS in perception.

Ablation Study We conduct an ablation study to show the effectiveness of various components in SSS. These results prove how each component contributes to the final performance improvement. We give details in the SM.

5. Conclusion, Discussion, and Future Work

We proposed Student Splatting and Scooping (SSS), a new non-monotonic mixture model, consisting of positive and negative Student's t distributions, learned by a principled SGHMC sampling. SSS contains a simple yet strong and non-trivial generalization of 3DGS and its variants. SSS outperforms existing methods in rendering quality, and shows high parameter efficiency, *e.g*. achieving comparable quality with less than $\frac{1}{3}$ of components.

SSS has limitations. Its primitives are restricted to symmetric and smooth t-distributions, limiting its representation. The sampling also needs hyperparameter tuning such as the percentage of negative components. In the future, we will combine other distribution families (*e.g*. Laplace) with t-distribution to further enhance the expressivity, and make the SGHMC self-adaptive to achieve better balances between the positive and negative components.

Acknowledgement

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 899739 CrowdDNA.

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. 5

[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. 2, 5, 7

[3] Guikun Chen and Wenguan Wang. A survey on 3d gaussian splatting. *arXiv preprint arXiv:2401.03890*, 2024. 1

[4] Tianqi Chen, Emily Fox, and Carlos Guestrin. Stochastic gradient hamiltonian monte carlo. In *International conference on machine learning*, pages 1683–1691. PMLR, 2014. 4

[5] Zhiqin Chen, Thomas Funkhouser, Peter Hedman, and Andrea Tagliasacchi. Mobilenerf: Exploiting the polygon rasterization pipeline for efficient neural field rendering on mobile architectures. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 16569–16578, 2023. 2

[6] Ben Fei, Jingyi Xu, Rui Zhang, Qingyuan Zhou, Weidong Yang, and Ying He. 3d gaussian as a new vision era: A survey. *arXiv preprint arXiv:2402.07181*, 2024. 1

[7] Sara Fridovich-Keil, Alex Yu, Matthew Tancik, Qinhong Chen, Benjamin Recht, and Angjoo Kanazawa. Plenoxels: Radiance fields without neural networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 5501–5510, 2022. 2

[8] 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, 2, 3, 5

[9] 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. 5, 7

[10] Peter Hedman, Pratul P Srinivasan, Ben Mildenhall, Jonathan T Barron, and Paul Debevec. Baking neural radiance fields for real-time view synthesis. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 5875–5884, 2021. 2

[11] Binbin Huang, Zehao Yu, Anpei Chen, Andreas Geiger, and Shenghua Gao. 2d gaussian splatting for geometrically accurate radiance fields. In *ACM SIGGRAPH 2024 Conference Papers*, pages 1–11, 2024. 2

[12] Letian Huang, Jiayang Bai, Jie Guo, and Yanwen Guo. Gs++: Error analyzing and optimal gaussian splatting. *arXiv preprint arXiv:2402.00752*, 2024. 2, 3

[13] 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, 2, 3, 5

[14] 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, 2, 3, 4, 5

[15] 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. 5, 7

[16] Jonas Kulhanek, Songyou Peng, Zuzana Kukelova, Marc Pollefeys, and Torsten Sattler. WildGaussians: 3D gaussian splatting in the wild. *NeurIPS*, 2024. 1

[17] Joo Chan Lee, Daniel Rho, Xiangyu Sun, Jong Hwan Ko, and Eunbyung Park. Compact 3d gaussian representation for radiance field. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 21719–21728, 2024. 1

[18] Haolin Li, Jinyang Liu, Mario Sznaier, and Octavia Camps. 3d-hgs: 3d half-gaussian splatting. *arXiv preprint arXiv:2406.02720*, 2024. 1, 2, 5

[19] Ruoshi Liu, Rundi Wu, Basile Van Hoorick, Pavel Tokmakov, Sergey Zakharov, and Carl Vondrick. Zero-1-to-3: Zero-shot one image to 3d object. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 9298–9309, 2023. 2

[20] Lorenzo Loconte, Aleksanteri Sladek, Stefan Mengel, Martin Trapp, Arno Solin, Nicolas Gillis, and Antonio Vergari. Subtractive mixture models via squaring: Representation and learning. In *International Conference on Learning Representations (ICLR)*, 2024. 1, 3

[21] 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. 2, 3, 5

[22] Ben Mildenhall, Pratul P Srinivasan, Matthew Tancik, Jonathan T Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. *Communications of the ACM*, 65(1):99–106, 2021. 2, 5

[23] Thomas Müller, Alex Evans, Christoph Schied, and Alexander Keller. Instant neural graphics primitives with a multiresolution hash encoding. *ACM transactions on graphics (TOG)*, 41(4):1–15, 2022. 2

[24] Ben Poole, Ajay Jain, Jonathan T Barron, and Ben Mildenhall. Dreamfusion: Text-to-3d using 2d diffusion. In *The Eleventh International Conference on Learning Representations*. 2

[25] Albert Pumarola, Enric Corona, Gerard Pons-Moll, and Francesc Moreno-Noguer. D-nerf: Neural radiance fields for dynamic scenes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10318–10327, 2021. 2

[26] Samuel Rota Bulò, Lorenzo Porzi, and Peter Kontschieder. Revising densification in gaussian splatting. In *European Conference on Computer Vision*, pages 347–362. Springer, 2024. 2, 5

[27] Steven M Seitz, Brian Curless, James Diebel, Daniel Scharstein, and Richard Szeliski. A comparison and evaluation of multi-view stereo reconstruction algorithms. In *2006 IEEE computer society conference on computer vision and pattern recognition (CVPR'06)*, pages 519–528. IEEE, 2006. 2

[28] Xiangyu Sun, Joo Chan Lee, Daniel Rho, Jong Hwan Ko, Usman Ali, and Eunbyung Park. F-3dgs: Factorized coordinates and representations for 3d gaussian splatting. In *Proceedings of the 32nd ACM International Conference on Multimedia*, pages 7957–7965, 2024. 1

[29] Jiaxiang Tang, Jiawei Ren, Hang Zhou, Ziwei Liu, and Gang Zeng. Dreamgaussian: Generative gaussian splatting for efficient 3d content creation. In *The Twelfth International Conference on Learning Representations*. 2

[30] Shimon Ullman. The interpretation of structure from motion. *Proceedings of the Royal Society of London. Series B. Biological Sciences*, 203(1153):405–426, 1979. 2

[31] 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. 5

[32] Max Welling and Yee W Teh. Bayesian learning via stochastic gradient langevin dynamics. In *Proceedings of the 28th international conference on machine learning (ICML-11)*, pages 681–688. Citeseer, 2011. 5

[33] Guanjun Wu, Taoran Yi, Jiemin Fang, Lingxi Xie, Xiaopeng Zhang, Wei Wei, Wenyu Liu, Qi Tian, and Xinggang Wang. 4d gaussian splatting for real-time dynamic scene rendering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 20310–20320, 2024. 2

[34] Minye Wu and Tinne Tuytelaars. Implicit gaussian splatting with efficient multi-level tri-plane representation. *arXiv preprint arXiv:2408.10041*, 2024. 1, 2

[35] Ziyi Yang, Xinyu Gao, Wen Zhou, Shaohui Jiao, Yuqing Zhang, and Xiaogang Jin. Deformable 3d gaussians for high-fidelity monocular dynamic scene reconstruction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 20331–20341, 2024. 2

[36] Zehao Yu, Anpei Chen, Binbin Huang, Torsten Sattler, and Andreas Geiger. Mip-splatting: Alias-free 3d gaussian splatting. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 19447–19456, 2024. 2, 3

[37] Jiahui Zhang, Fangneng Zhan, Muyu Xu, Shijian Lu, and Eric Xing. Fregs: 3d gaussian splatting with progressive frequency regularization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 21424–21433, 2024. 2, 3, 5

[38] Kai Zhang, Gernot Riegler, Noah Snavely, and Vladlen Koltun. Nerf++: Analyzing and improving neural radiance fields. *arXiv preprint arXiv:2010.07492*, 2020. 2

[39] 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. 5

[40] Matthias Zwicker, Hanspeter Pfister, Jeroen Van Baar, and Markus Gross. Ewa volume splatting. In *Proceedings Visualization, 2001. VIS'01.*, pages 29–538. IEEE, 2001. 2, 3

[41] Matthias Zwicker, Hanspeter Pfister, Jeroen Van Baar, and Markus Gross. Ewa splatting. *IEEE Transactions on Visualization and Computer Graphics*, 8(3):223–238, 2002. 2, 3