EVSplitting: An Efficient and Visually Consistent Splitting Algorithm for 3D Gaussian Splatting

QI-YUAN FENG, BNRist, Department of Computer Science and Technology, Tsinghua University, China
GENG-CHEN CAO, BNRist, Department of Computer Science and Technology, Tsinghua University, China
HAO-XIANG CHEN, BNRist, Department of Computer Science and Technology, Tsinghua University, China
QUN-CE XU*, BNRist, Department of Computer Science and Technology, Tsinghua University, China
TAI-JIANG MU, BNRist, Department of Computer Science and Technology, Tsinghua University, China
RALPH MARTIN, School of Computer Science and Informatics, Cardiff University, United Kingdom
SHI-MIN HU, BNRist, Department of Computer Science and Technology, Tsinghua University, China

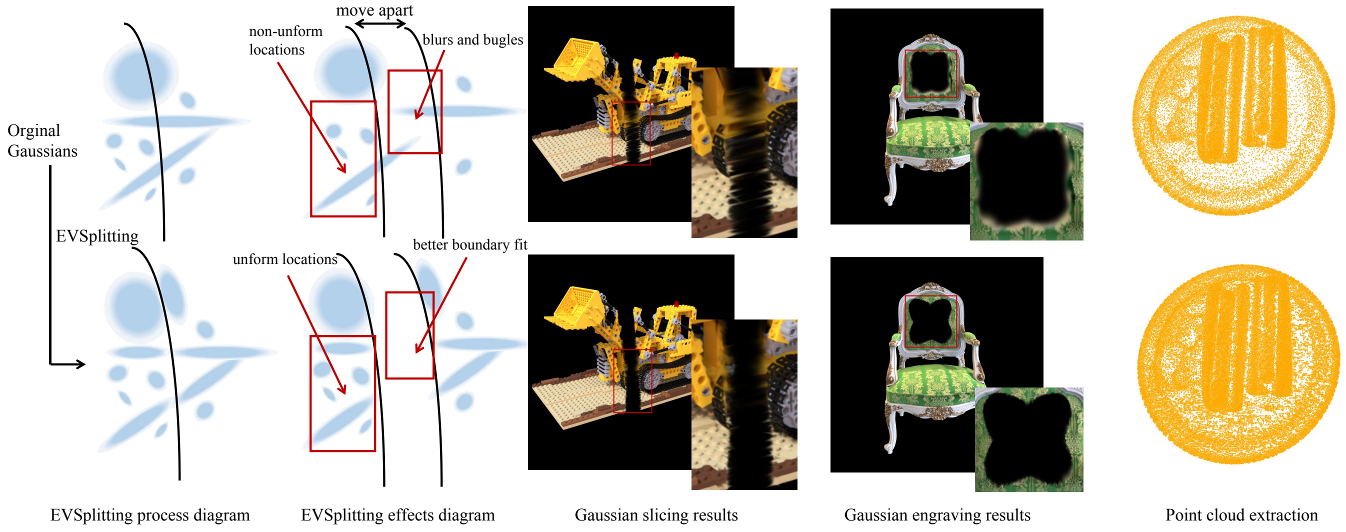

Fig. 1. Splitting operation is needed for learning-free and explicit operations on 3D Gaussian Splatting; (Top row) however, the straightforward splitting, i.e., simply keeping or rejecting the Gaussians intersected by the splitting plane, will produce bulges (the third column), blurs (the fourth column) and non-uniform distribution of locations (the last column). (Bottom row) These problems can be solved with our EVSplitting process (the first column), making Gaussians uniform and fitting the given boundary (the second column). EVSplitting can also benefit many other applications such as part extraction, texturing, etc.

This paper presents *EVSplitting*, an efficient and visually consistent splitting algorithm for 3D Gaussian Splatting (3DGS). It is designed to make operating 3DGS as easy and effective as other 3D explicit representations, readily for industrial productions. The challenges of above target are: 1) The huge number and complex attributes of 3DGS make it tough to explicitly operate on 3DGS in a real-time and learning-free manner; 2) The visual effect of 3DGS is very difficult to maintain during explicit operations and 3) The anisotropism of Gaussian always leads to blurs and artifacts. As far as we know, no prior work can address these challenges well. In this work, we introduce a direct and efficient 3DGS splitting algorithm to solve them. Specifically, we formulate the 3DGS splitting as two minimization problems that aim to ensure visual consistency and reduce Gaussian overflow across boundary (splitting plane), respectively. Firstly, we impose conservations on the zero-, first- and second-order moments of the weighted Gaussian distribution to guarantee visual consistency. Secondly, we reduce the boundary overflow with a special constraint on the aforementioned conservations.

[*]Corresponding author: Qun-Ce Xu (quncexu@tsinghua.edu.cn).

Authors' Contact Information: Qi-Yuan Feng, BNRist, Department of Computer Science and Technology, Tsinghua University, China, fqy22@mails.tsinghua.edu.cn; Geng-Chen Cao, BNRist, Department of Computer Science and Technology, Tsinghua University, China, cgc21@mails.tsinghua.edu.cn; Hao-Xiang Chen, BNRist, Department of Computer Science and Technology, Tsinghua University, China, chx20@mails.tsinghua.edu.cn; Qun-Ce Xu, BNRist, Department of Computer Science and Technology, Tsinghua University, China, quncexu@tsinghua.edu.cn; Tai-Jiang Mu, BNRist, Department of Computer Science and Technology, Tsinghua University, China, taijiang@tsinghua.edu.cn; Ralph Martin, School of Computer Science and Informatics, Cardiff University, United Kingdom, martinrr@cardiff.ac.uk; Shi-Min Hu, BNRist, Department of Computer Science and Technology, Tsinghua University, China, shimin@tsinghua.edu.cn.

SA Conference Papers '24, December 03–06, 2024, Tokyo, Japan

© 2024 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-1131-2/24/12
https://doi.org/10.1145/3680528.3687592

With these conservations and constraints, we derive a closed-form solution for the 3DGS splitting problem. This yields an easy-to-implement, plug-and-play, efficient and fundamental tool, benefiting various downstream applications of 3DGS.

CCS Concepts: • **Computing methodologies → Point-based models**; • **Mathematics of computing → Probabilistic algorithms**.

Additional Key Words and Phrases: 3D reconstruction, Gaussian splatting, neural rendering, 3D representation, editing

ACM Reference Format:
Qi-Yuan Feng, Geng-Chen Cao, Hao-Xiang Chen, Qun-Ce Xu, Tai-Jiang Mu, Ralph Martin, and Shi-Min Hu. 2024. EVSplitting: An Efficient and Visually Consistent Splitting Algorithm for 3D Gaussian Splatting. In *SIGGRAPH Asia 2024 Conference Papers (SA Conference Papers '24), December 03–06, 2024, Tokyo, Japan.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3680528.3687592

1 introduction

A suitable 3D representation is crucial in artistic model design, special effects production, and VR/AR applications. While some implicit 3D representations, such as neural radiance fields (NeRFs) [Mildenhall et al. 2020], have achieved excellent results in rendering quality, they are difficult to edit and slow to render. Consequently, most designers and modeling artists prefer explicit representations, particularly point clouds and meshes, which can be quickly rendered and easily edited. 3D Gaussian splatting (3DGS) [Kerbl et al. 2023] is a novel 3D explicit representation that utilizes a set of 3D positions, opacity, anisotropic covariance, and spherical harmonic (SH) coefficients to represent a 3D scene. 3DGS can be rapidly rendered using a differentiable rasterization. This representation has been widely used for learning-based 3D editing [Chen et al. 2023a; Fang et al. 2023; Lan et al. 2023] and 3D geometry generation [Chung et al. 2023; Tang et al. 2024b]. Previous works mostly concentrate on the novel view synthesis functionality of 3DGS. However, the real-time, learning-free operations on the 3DGS should be further researched to make it more applicable, flexible and compatible for industrial production, such as part/object extraction, engraving, texturing, subdivision, conversion, etc.

The above operations in industrial production usually involve splitting the 3DGS. For example, to engrave a 3DGS, one should split the 3DGS along the cutting curve and discard the undesired Gaussians. Note that two goals are pursued in splitting a 3DGS: i) the overall visual consistency should be kept and ii) the Gaussian overflow across the boundary (splitting plane) should be avoided. However, the straightforward splitting, which simply keeps or rejects the Gaussians intersected by the splitting plane, would suffer from bugles, blurs and holes in the results as shown in Fig. 1. Even worse, a sequence of such editing operations can lead to the collapse of the entire 3DGS model. Although Gaussian mixture model (GMM) [Moon 1996] can approximate the splitting process of a Gaussian distribution. It is inefficient due to non-parallel for multiple fitted curves and iterative optimization nature, making it inapplicable for real-time applications, especially for 3DGS, which contains tens of thousands of Gaussian kernels.

Inspired by Hall et al. [2002], who provided an algorithm to add and subtract eigenspaces using eigenvalue decomposition and singular value decomposition under rank conservation, we present *EVSplitting*, an efficient and effective splitting algorithm for 3DGS as shown in Fig. 2 by exploring novel conservations. We formulate the 3DGS splitting as two minimization problems, aiming at ensuring the visual consistency during splitting and avoiding split Gaussians overflowing across the boundary, respectively. We solve the above two minimization problems by conducting the following constraints: 1) conserve the zero-, first-, and second-order moments of the weighted Gaussian distribution; 2) derive special variants of the moments conversations adhering to the minimization of Gaussian overflow; 3) combine the variants with some inherent characteristics of Gaussian distribution to ensure the uniqueness of the solution. This process allows us to model the Gaussian splitting via a set of integral tensor equations, which have an efficient and visually consistent closed-form solution.

We further demonstrate the benefits of EVSplitting across various applications. For Gaussian-splatting-based explicit editing and many other applications, our method produces clearer boundaries and consistent visual appearance. For point cloud extraction from the 3DGS model, our method enables flat, texture-less regions to be represented by more homogeneous Gaussians, leading to a denser and more uniform point cloud.

In summary, this paper makes the following contributions:

- We formulate Gaussian splitting as minimization problems with energy functions for ensuring visual consistency and avoiding Gaussian overflow.
- We derive a closed-form solution with variants of the energy functions and convert it to an efficient split algorithm, dubbed as EVSplitting, which is applicable to any 3DGS.
- We demonstrate how our EVSplitting benefits various applications, including part/object extraction, engraving, texturing and point cloud extraction.

2 Related work

2.1 3D Representations

A 3D representation is a data structure for handling and storing 3D geometry. There are many different kinds of 3D representations, each with its own advantages and disadvantages. These representations can be divided into explicit, implicit and hybrid categories. Implicit representations include NeRFs [Barron et al. 2022; Mildenhall et al. 2020; Müller et al. 2022],NeuS [Wang et al. 2021], iso-surfaces [Zhang et al. 2023],multiscale hash encoding[Deng et al. 2024], etc., supporting high-quality novel view synthesis and 3D content generation from text [Poole et al. 2023; Wang et al. 2023b] or images [Sun et al. 2024] due to its flexible geometry. However, they are very difficult to be edited because of their implicit nature. Explicit representations such as point cloud and mesh, along with well-developed editing and processing tools [Himmelsbach et al. 2010; Sorkine and Alexa 2007], have been widely used in industrial production. Hybrid representations like [Guo et al. 2023; Shen et al. 2021; Xu et al. 2022] try to merge the explicit representation with implicit fields, for faster rendering and flexibility of geometry; they are still inconvenient to be edited, as some attributes are represented with implicit fields.

2.2 3DGS and Its Applications

3DGS [Kerbl et al. 2023] proposes a new explicit 3D representation, which is convenient to obtain from images with machine learning techniques. Thanks to the explicit nature, 3DGS supports real-time, high-quality image rendering, and draws widespread interest in many areas, such as Simultaneous Localization And Mapping (SLAM), dynamic scene rendering, and Artificial Intelligence Generated Content (AIGC). SLAM researchers [Huang et al. 2023; Matsuki et al. 2023] employ 3DGS as landmarks and design special tracking and Gaussian optimization strategy, showing a huge improvement in view synthesis. 3DGS is also extended as 4DGS to model the dynamic scene [Wu et al. 2023; Yang et al. 2024]. AIGC researchers explore the potential of 3DGS-based generation [Chen et al. 2023b] by distilling image diffusion prior [Ho et al. 2020] or large reconstruction model [Tang et al. 2024a]. Though 3DGS can be edited by the guidance of diffusion models[Chen et al. 2023a; Fang et al. 2023; Lan et al. 2023], but there are no learning-free and real-time explicit editing operations on 3DGS as far as we know. With 3DGS, the model's texture and geometry are strongly related. When engraving shapes or modifying the texture, we need to split the Gaussians adhering to the boundaries of shapes or texture. We provide a new splitting method for 3DGS to generate more visually consistent appearance and clearer boundaries during explicit geometry processing, thus benefiting such applications.

3 Efficient and Visually Consistent 3DGS Splitting

Our goal is to split the original 3DGS efficiently with high visual consistency and low Gaussian overflow across the splitting plane. We first briefly overview the 3DGS representation in Sec. 3.1. Then we define the splitting problem in Sec. 3.2. We introduce some conservation rules used in the splitting process and explain their physical meaning in Sec. 3.3. A closed-form solution to the splitting problem is derived in Sec. 3.4, and we consider implementation details in Sec. 3.5.

3.1 3D Gaussian Splatting Representation

A 3DGS model represents a 3D target using a set of 3D Gaussians, each characterized by its 3D position μ, opacity α, covariance Σ, and spherical harmonic (SH) coefficients. A 3D point with a coordinate $\mathbf{x}$ covered by a 3D Gaussian k obtains its opacity $\alpha(\mathbf{x})$ from that Gaussian as below (unless specifically stated, the opacity α is not normalized in this paper for brevity of derivation—in the actual implementation):

$$\alpha(\mathbf{x}) = \alpha_k \, \mathrm{pdf}_k(\mathbf{x}) \tag{1}$$

$$\mathrm{pdf}_k(\mathbf{x}) = \frac{1}{(2\pi)^{3/2}\sqrt{|\Sigma_k|}} \exp(-\frac{1}{2}(\mathbf{x}-\mu_k)^T \Sigma_k^{-1}(\mathbf{x}-\mu_k)) \tag{2}$$

where $\mathrm{pdf_k}$ denotes the probability density function of Gaussian k. Due to the positive definiteness of covariance Σ, it can be decomposed as follows to save storage space:

$$\Sigma = RSS^T R^T \tag{3}$$

where R is a rotation matrix and S is a diagonal matrix of scalings in the principal directions. R can be stored as a quaternion q. All of

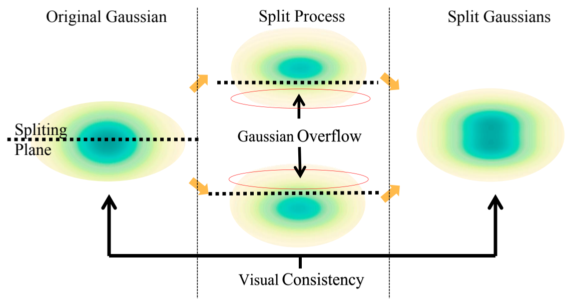

Fig. 2. Gaussian splitting problem (2D example) aims to minimize the Gaussian overflow while maximizing the visual consistency. The cross-boundary part (in red circles in the second column) indicates the Gaussian overflow. Visual consistency means the visual similarity between the initial Gaussian (the first column) and the split Gaussians (the third column).

these can be converted to respective matrices and combined, making sure to normalize q to obtain a valid unit quaternion.

The rasterization process projects Gaussians into the image plane and utilizes the alpha blending strategy to get the color. In alpha blending, the final color is a weighted average of projected Gaussians, and the weight is computed by $w_i = (\alpha_{gi}\Pi_{j=1}^{i-1}(1-\alpha_{gj}))$ where α_{gi} means the opacity of the i-th nearest Gaussian in the view direction.

3.2 Problem Definition for Gaussian Splitting

Since the scale matrix S is square and diagonal, the eigenvalue matrix is also square and diagonal, and can be represented by:

$$\Lambda = SS^T \tag{4}$$

where Λ is a diagonal matrix whose elements are the squared scales of the principal components. Each column vector of R represents the direction vector of a principal component. Since the covariance matrix is a symmetric positive matrix, the three principal component direction vectors are mutually perpendicular unit vectors.

The explicit operations, like engraving, texturing, etc., on 3DGS should be efficient for interaction, so the split operation illustrated in Fig. 2 should not rely on further training. Moreover, if visual consistency cannot be guaranteed and heavy Gaussian overflow across the splitting plane occurs in such processing, artifacts will accumulate during the editing and extraction steps.

Preventing overflow involves ensuring that the principal component does not become too long and exceeds the splitting boundary. Therefore, our problem is how to split the principal components of a 3D Gaussian to fit the boundaries while maintaining its 3D visual consistency. Specifically, we simplify this problem to splitting Gaussians with a plane, as any complex curve can be approximately represented by splitting with many planes. Therefore, the problem above is reduced to how to split a 3D Gaussian using a plane while retaining 3D visual consistency. A plane can be expressed as:

$$P(\mathbf{n}, d) = \{\mathbf{x} \in \mathbb{R}^3 | \hat{P}(\mathbf{n}, d)(\mathbf{x}) \triangleq \mathbf{n} \cdot \mathbf{x} + d = 0\} \tag{5}$$

where $\mathbf{n}$ denotes the unit normal of the plane, d is its distance from the origin and $\hat{P}$ denotes directed distance. We then formally define the splitting at the split plane P of a Gaussian $(\alpha_0, \mu_0, \Sigma_0)$ into two new (left l and right r) Gaussians $(\alpha_l, \mu_l, \Sigma_l)$ and $(\alpha_r, \mu_r, \Sigma_r)$ as:

$$\alpha_l, \alpha_r, \mu_l, \mu_r, \Sigma_l, \Sigma_r = F(\alpha_0, \mu_0, \Sigma_0, P(\mathbf{n}, d)) \tag{6}$$

Note that in actual implementation, α_0 is the denormalized opacity of the original Gaussian, and α_l and α_r should be normalized to the final split Gaussians.

As illustrated in Fig. 2, this process aims to minimize the intersection between the two new Gaussians (i.e., Gaussian overflow) while maximizing the similarity between each new Gaussian and its own part of the original (i.e., visual consistency). The visual consistency performs as the unchanged rendering results. Consider that the alpha blending process in 3DGS is an approximately of real volume rendering, the unchanged $\alpha(\mathbf{x})$ in Eq. 1 results in unchanged density in volume rendering. Ideally, one would require $\alpha(\mathbf{x})$ to remain unchanged for all position $\mathbf{x}$. So, the minimization energy is as follows:

$$\underset{\alpha_k, \Sigma_k, \mu_k}{\operatorname{argmin}} \int_{R^3 \setminus \mathbb{V}_k} \alpha_k \mathrm{pdf}_k(\mathbf{v}) dV, \quad k \in \{l, r\} \tag{7}$$

$$\underset{\alpha_k, \Sigma_k, \mu_k}{\operatorname{argmin}} \left| \int_{\mathbb{V}_k} \alpha_k \mathrm{pdf}_k(\mathbf{v}) dV - \int_{\mathbb{V}_k} \alpha_0 \mathrm{pdf}_0(\mathbf{v}) dV \right|, \quad k \in \{l, r\} \tag{8}$$

where $\mathbf{v}$ is the vector corresponding to spatial integrating unit V, the pdf comes from Eq. 2 and $\mathbb{V}_k$ is the half-space on the appropriate side of the plane:

$$\mathbb{V}_l = \{\mathbf{x} \in \mathbb{R}^3 | \hat{P}(\mathbf{n}, d)(\mathbf{x}) < 0\} \tag{9}$$

$$\mathbb{V}_r = \{\mathbf{x} \in \mathbb{R}^3 | \hat{P}(\mathbf{n}, d)(\mathbf{x}) \geq 0\} \tag{10}$$

The Eqs.7 aims to minimize the volume of 3D Gaussians exceeding the boundary while the Eq.8 aims to minimize the volume changes in the reserve part. Because 3D Gaussians have a fixed shape, it is impossible to minimize both Eq. 7 and 8 simultaneously. Thus, we should consider them together when evaluating 3D visual consistency and extent beyond the boundary. Therefore, we define interval error E_i and external excess E_e according to Eqs. 7 and 8, respectively.

$$E_e = \frac{1}{|\mathbb{G}|} \sum_{g \in \mathbb{G}} \sum_{k \in \{l, r\}} \int_{R^3 \setminus \mathbb{V}_k} \alpha_k \mathrm{pdf}_k(\mathbf{v}) dV \tag{11}$$

$$E_i = \frac{1}{|\mathbb{G}|} \sum_{g \in \mathbb{G}} \sum_{k \in \{l, r\}} \left| \int_{\mathbb{V}_k} \alpha_k \mathrm{pdf}_k(\mathbf{v}) dV - \int_{\mathbb{V}_k} \alpha_g \mathrm{pdf}_g(\mathbf{v}) dV \right| \tag{12}$$

where $\mathbb{G}$ is the entire Gaussian model to be split and $\mathbb{V}_k$ is the half-space where the split 3D Gaussian k is located.

In summary, reducing E_i leads to new Gaussians with visual effects more similar to those of the original Gaussian while reducing E_e results in new Gaussians that better fit the border, with fewer blurs and needle-like bulges. Both of the E_i and E_e affect the split results.

3.3 Conservation and Constraints

Ensuring visual consistency is more important than reducing exceeding boundaries in preventing model structure collapse. Therefore, we should conserve the opacity distribution $\alpha(x)$ in Eq. 2 as much as possible during splitting as the following three constraints. To ensure that the new Gaussian looks the same as the original one from any perspective to achieve a smaller E_i in Eq. 12, the cumulative of local opacity $\alpha(x)$ should be conserved, i.e., the zero-order moment:

$$\int_{\mathbb{R}^3} \alpha_0 \mathrm{pdf}_0(\mathbf{v}) dV = \int_{\mathbb{R}^3} \alpha_l \mathrm{pdf}_l(\mathbf{v}) dV + \int_{\mathbb{R}^3} \alpha_r \mathrm{pdf}_r(\mathbf{v}) dV \tag{13}$$

Similarly, to ensure that the center of the local opacity distribution remains visually unchanged for both the original Gaussian and the split Gaussians, we should conserve the first-order moment:

$$\int_{\mathbb{R}^3} \alpha_0 \mathbf{v} \mathrm{pdf}_0(\mathbf{v}) dV = \int_{\mathbb{R}^3} \alpha_l \mathbf{v} \mathrm{pdf}_l(\mathbf{v}) dV + \int_{\mathbb{R}^3} \alpha_r \mathbf{v} \mathrm{pdf}_r(\mathbf{v}) dV \tag{14}$$

For each point, to ensure that the scale of the local opacity distribution remains as consistent as possible, we aim to keep the mean of the outer product matrix unchanged. This requires the conservation of the second-order moment:

$$\int_{\mathbb{R}^3} \alpha_0 \mathbf{v}\mathbf{v}^T \mathrm{pdf}_0(\mathbf{v}) dV = \int_{\mathbb{R}^3} \alpha_l \mathbf{v}\mathbf{v}^T \mathrm{pdf}_l(\mathbf{v}) dV + \int_{\mathbb{R}^3} \alpha_r \mathbf{v}\mathbf{v}^T \mathrm{pdf}_r(\mathbf{v}) dV \tag{15}$$

Via these conserved quantities, we can reduce the splitting problem to an optimization problem constrained by this integral tensor equation set (ITEs).

3.4 Closed Form Solutions for Gaussian Splitting

The Eqs. (13-15) have more than 11 variables$(\alpha_l, \alpha_r, \mu_\mathbf{l}, \mu_\mathbf{r}, \Sigma_\mathbf{l}, \Sigma_\mathbf{r})$, but there are only 11 equations in total.This makes it an underdetermined system with infinitely many solutions. We still need to find a solution that minimizes E_e in Eq. 11 under these constraints. A special solution exists by additionally requiring the two Gaussians to be separated in space (giving a local minimal E_e), while ensuring visual consistency of the left and right sides separately (giving a low E_i as explained in Sec. 3.3):

$$\int_{\mathbb{V}_k} \alpha_0 \omega(\mathbf{v}) \mathrm{pdf}_0(\mathbf{v}) dV = \int_{\mathbb{R}^3} \alpha_k \omega(\mathbf{v}) \mathrm{pdf}_k(\mathbf{v}) dV, \tag{16}$$

$$where\ k \in \{l, r\}, \omega(\mathbf{v}) \in \{\mathbf{1}, \mathbf{v}, \mathbf{v}\mathbf{v}^T\} \tag{17}$$

These equations split the Eqs. (13-15) to maintain characteristics conservation on both sides, respectively. Any 3D Gaussian i satisfies the following equation concerning its second central moment:

$$\Sigma_i + \mu_i \mu_i^T = \int_{\mathbb{R}^3} \mathbf{v}\mathbf{v}^T \mathrm{pdf}_i(\mathbf{v}) dV \tag{18}$$

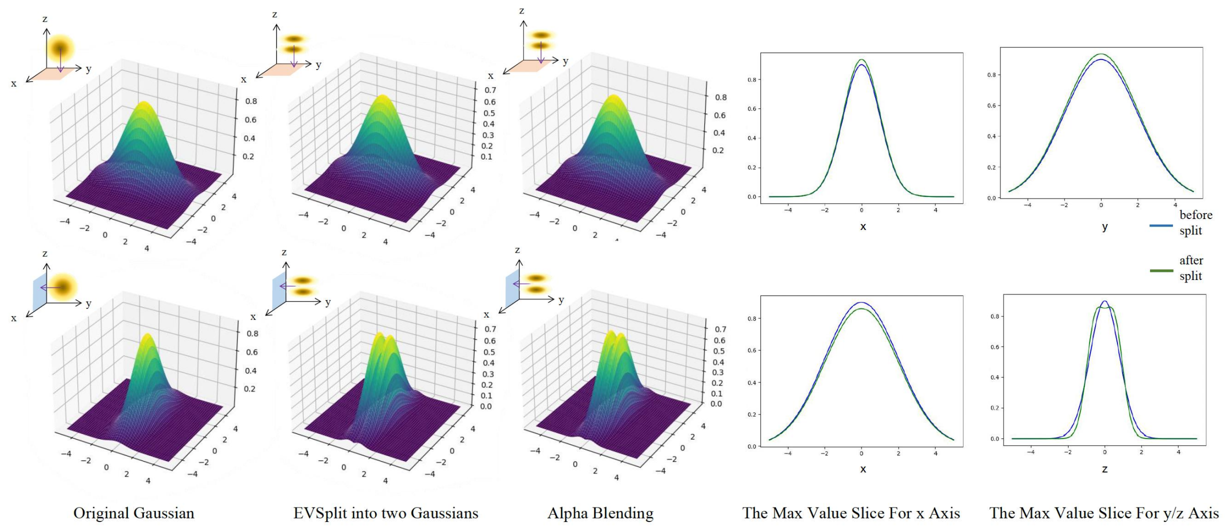

Fig. 3. A 3DGS with only one Gaussian and rasterization example. The GS model uses alpha blending[Porter and Duff 1984] strategy when rendering views. The result is represented by projection, similar to the rasterization process of 3DGS[Kerbl et al. 2023]. We display the xy(top) and xz(bottom) image plane. The first column shows the alpha blending results of the original single Gaussian, the second column shows the split results and the third column shows the alpha blending results of the split Gaussians. The last two columns display the maximum value slices of the alpha blending results from the first and the third columns in different axes. Notably, there is no obvious difference between the alpha blending results of the original and the split Gaussians.

Combining this with Eqs. (16,17), we can derive the closed-form solutions for the split equations below:

$$\alpha_k = \alpha_0 C_k, k \in \{l, r\} \tag{19}$$

$$\mu_l = \mu_0 - \frac{\mathbf{L}_0 D}{\tau C_l} \tag{20}$$

$$\mu_r = \mu_0 + \frac{\mathbf{L}_0 D}{\tau C_r} \tag{21}$$

$$\Sigma_l = \Sigma_0 + \frac{\mathbf{L}_0 \mathbf{L}_0^T}{\tau^2}\left(\frac{d_0 D}{\tau C_l} - \frac{D^2}{C_l^2}\right) \tag{22}$$

$$\Sigma_r = \Sigma_0 - \frac{\mathbf{L}_0 \mathbf{L}_0^T}{\tau^2}\left(\frac{d_0 D}{\tau C_r} + \frac{D^2}{C_r^2}\right) \tag{23}$$

where:

$$C_l = \frac{1}{2}\left(1 - \operatorname{erf}\left(\frac{d_0}{\sqrt{2}\tau}\right)\right) \tag{24}$$

$$C_r = \frac{1}{2}\left(1 + \operatorname{erf}\left(\frac{d_0}{\sqrt{2}\tau}\right)\right) \tag{25}$$

$$D = \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{d_0^2}{2\tau^2}\right) \tag{26}$$

$$\mathbf{L}_0 = \Sigma_0 \mathbf{n} \tag{27}$$

$$\tau = \sqrt{\mathbf{n}^T \Sigma_0 \mathbf{n}} \tag{28}$$

$$d_0 = \hat{P}(\mathbf{n}, d)(\mu_0) \tag{29}$$

In the above, erf is the error function from probability theory, and d_0 is the distance from the Gaussian position to the split plane. Detailed derivations are provided in the Supplementary Material. For a 3DGS model, the color is computed based on the view direction (from the camera position to the Gaussian position). The distance between the new Gaussians and the original Gaussian is less than the scale of the original Gaussian. In practical applications, the scale of the Gaussian is much smaller than the distance between the camera and the Gaussian, so the split operation has almost no effect on the view direction. Therefore, we just copy the SH coefficient in the two new Gaussians, which simplifies the calculation.

Table 1. Comparison of the effectiveness of reducing inhomogeneity during training between EVSPlitting and intuitively restricted to isotropy.

Reducing Strategy	*PSNR* ↓	*SSIM* ↓	*LPIPS* ↑
EVSplitting	36.75	0.98	0.0238
Restricted to isotropy	34.72	0.97	0.0398

3.5 Implementation Details for Splitting

Due to floating point precision issues and the properties of Gaussian distributions, errors will accumulate as splitting continues (e.g. when multiple edits are performed). We use the following approach to compensate for these problems. Firstly, if the split plane $P(\mathbf{n}, d)$ is far from the position of the original Gaussian, the split would be unnecessary, because it creates a Gaussian that is very similar to the original one and another Gaussian that is almost invisible due to its very small α. Furthermore, if splitting is applied to every Gaussian, there will be a huge amount of needless calculation. Inspired by the Gaussian filter[Kerbl et al. 2023]'s influence range threshold, i.e, $3 \times \max(\mathbf{Tri}(\mathbf{S}))$ (where $\mathbf{Tri}$ denotes the main diagonal vector), we choose a threshold η for $|d_0|$:

$$\eta = 3 \times \max((R \cdot \mathbf{n}) \odot \mathbf{Tri}(S)) \tag{30}$$

If $|d_0| < \eta$, we split the Gaussian with the plane; otherwise, we leave the original Gaussian unchanged.

Secondly, the weight denominators C_l and C_r may be very small, compromising floating point accuracy. Thus we add an offset $\epsilon = 10^{-20}$ to C_l, C_r and D:

$$C_l = \frac{1}{2}(1 - \text{erf}(\frac{d_0}{\sqrt{2}\tau}) + \epsilon) \tag{31}$$

$$C_r = \frac{1}{2}(1 + \text{erf}(\frac{d_0}{\sqrt{2}\tau}) + \epsilon) \tag{32}$$

$$D = \frac{1}{\sqrt{2\pi}}(\exp(-\frac{d_0^2}{2\tau^2}) + \epsilon) \tag{33}$$

Thirdly, floating point precision issues may cause the eigenvalue matrix to deviate slightly from being a real positive definite matrix. We correct any such issues in our eigenvalue matrix Λ_i by replacing it with Λ_{ti}, where:

$$\Lambda_{ti} = \text{ReLU}(\Lambda_i), \quad i \in \mathbb{G} \tag{34}$$

Fourthly, in computer graphics, a right-handed coordinate system is typically used. A rotation matrix R defining a left-handed space cannot be transformed into a right-handed coordinate system by a normalized quaternion (the improper rotation problem). Since a 3D Gaussian is completely centrally symmetric, we can turn the left-handed rotation matrix R_i into a right-handed rotation matrix R_{ti} by computing:

$$R_{ti} = R_i \det(R_i), \quad i \in \mathbb{G} \tag{35}$$

4 Experiments

To demonstrate the effectiveness of our splitting algorithm, we apply it to explicit GS model editing, part/object extraction, engraving, texture mapping and point cloud uniformization, and compare EVSplitting to other solutions.

4.1 Basic Split and Its Applications

To illustrate the efficiency and accuracy of EVSplitting, we performed the following experiments.

Firstly, we demonstrate the core idea of EVSplitting with a simple 3DGS example using a single 3D Gaussian. Fig. 3 shows the process of splitting this original kernel into two along the xy plane. We provide alpha blending results from views parallel and perpendicular to the split plane, rasterizing them to the respective view planes. The curves before and after the split are similar, demonstrating that our split algorithm maintains the visual appearance.

Secondly, we apply our splitting algorithm to remove inhomogeneous Gaussians (here, the threshold ratio of the largest to the smallest scale exceeds 5) in the original 3DGS model from the NeRF synthetic dataset [Mildenhall et al. 2020], as shown in Fig. 4. We achieve an almost identical visual appearance after the split. To illustrate that our method is better than intuitively restricting Gaussians to isotropy, we test our EVSplitting and the 3D isotropic Gaussian Splatting in NeRF Synthetic dataset and report the quantitative results in Table 1.

Table 2. Plane splitting quality measured by E_i and E_e on the Nerf synthetic[Mildenhall et al. 2020] dataset

Split Strategy	$E_i \downarrow$	$E_e \downarrow$
Naive solution	66.6819	0.01056
Move solution	0.0000	2.3823
Remove solution	15.2541	0.0000
EVSplitting	2.3819	0.0003

Table 3. Plane splitting quality measured by E_i and E_e on the Mip-NeRF 360[Barron et al. 2022] dataset

Split Strategy	$E_i \downarrow$	$E_e \downarrow$
Naive solution	107.9933	1.1193
Move solution	0.0000	12.3163
Remove solution	84.4767	0.0000
EVSplitting	11.9186	0.3981

Thirdly, a Gaussian mixture model (GMM) [Moon 1996] can be also used to split Gaussians. We use EVSplitting and GGMMs (GMMs with GPU acceleration) to fit the same object on an NVIDIA GeForce RTX 4090 GPU. For a 3-dimensional Gaussian split task with 100 random Gaussians, EVSplitting takes only 0.006 s while using GGMMs(steps = 20) takes about 1.399s, respectively. For EVSplitting , the average E_i and E_e are 0.0043 and 0.0095 while GGMMs's errors are 0.0038 and 0.0995. Furthermore, the GMMs algorithm has poor parallelism and is difficult to use in a 3DGS model with tens of thousands of points. We also trained an MLP (5 layers * 256 dims) to split the same scene. The training time is about 2h, and the inference time is about 0.217s. Quantitatively, the average E_i and E_e of this MLP are 0.0105 and 0.1200, respectively.

Fourthly, to quantitatively evaluate EVSplitting, we introduce "plane split", a basic operation that splits a 3DGS model with a 3D plane $P(\mathbf{n}, d)$ and move them apart. Since EVSplitting yields a GS model after the split, it can be applied several times to achieve better results. We evaluate the average performance on the entire NeRF synthetic dataset[Mildenhall et al. 2020], by splitting each 3DGS model with a randomly generated plane. We also compare EVSplitting to three other baselines:(1) the Naive solution, which intuitively copies, shrinks and then moves the Gaussians to the side their poistions are on; (2) the Move solution, which aims at the global minimum of E_i, directly moving Gaussians to the side their positions are on, and (3) the Remove solution, which aims at the global minimum of E_j, further removing any Gaussians intersecting the split plane. Quantitative results are presented in Table (2,3) and some qualitative results are presented in the three examples of Fig. 5. Additional results can be found in Fig. 9. EVSplitting achieves a small value for both indicators and ensures a good compromise between voids, blurring and needles. Note that it is impossible for both metrics to be smaller than Move and Remove solution simultaneously, due to the unimodal and symmetric shape of the Gaussian distribution. We also test the total execution and moving time of different solutions for plane splitting on an NVIDIA GeForce RTX

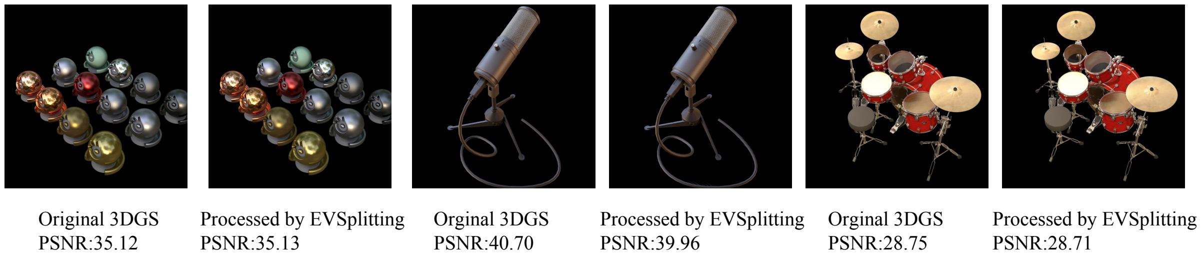

Fig. 4. A 3DGS model before and after inhomogeneity removal using EVSplitting . There is almost no visual difference between them.

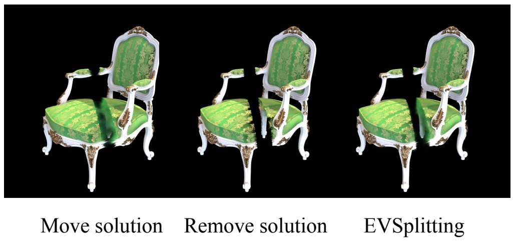

Fig. 5. In the above three examples, we split a chair into two parts along a randomly chosen plane and move them apart vertically to the split plane. EVSplitting produces sharp and smooth boundaries while maintaining visual consistency, compared with other solutions.

4090 GPU. The EVSplittings solution takes approximately 0.52s, compared to the Naive solution's 0.48s, the Move solution's 0.37s, and the Remove solution's 0.43s.

Plane splitting serves as a basic operation in many applications. A straightforward extension is to generate a slice image for 3D reconstruction [Wang et al. 2023a]. As shown in Figs. 5 and 10, our plane split can reduce the effects of inhomogeneous Gaussians in the model and obtain smoother split planes and curves, which is the requirement of slice image.

Another common tool is to extract or remove some specific parts/components from an existing model for purposes such as re-assembly. Given a 3DGS model, we can first use point cloud segmentation methods [Guo et al. 2021] to segment the parts/components based on the Gaussian coordinates. Then we find the closest point pair for neighboring parts to determine the split planes. Finally, we can achieve a smooth boundary and improved segmentation effects by applying our split operation using these splitting pairs. Some results are shown in Figs. (5, 11).

We also apply our algorithm on a real-world dataset, i.e., mipN-eRF360 [Barron et al. 2022], to segment out the foreground from the background using a boundingbox hint. We measure the decomposition accuracy using 3D-Intersection-over-Union (3DIoU, higher is better) of the voxels occupied by the decomposed object and the boundingbox. We compare our EVSplitting with the Filter solution that simply retains Gaussians positioned within the boundingbox. Besides, we also render 2D masks of the decomposed object and test the 2D image IoU results with SAM [Kirillov et al. 2023] masks of the object. Results are shown in Table 4. Although the inaccuracy

Table 4. The 3D and 2D IoU of the segmented foreground objects in mip-nerf 360 [Barron et al. 2022] scenes.

Scene	vase in garden		lego in kitchen	
Metric	3DIoU↑	2DIoU↑	3DIoU↑	2DIoU↑
EVSplitting	94.22%	62.35%	85.17%	43.56%
Filter solution	90.16%	59.33%	78.40%	39.46%

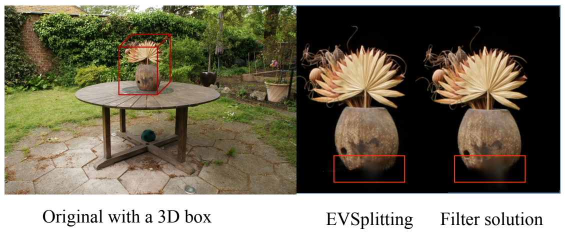

Fig. 6. Given a 3D foreground boundingbox hint, we can decompose the foreground from the background with our EVSplitting, producing foreground objects with fewer artifacts compared with the Filter solution.

of SAM limits the upperbound, the significant improvement in IoU for the entire object indicates a clear enhancement in boundary accuracy. The 3D/2D evaluation results demonstrate our superiority over the baseline method, proving the potential of EVSplitting for real-world applications.

4.2 Engraving and Texture Mapping

Engraving and texturing a 3DGS model are also common tools for geometric modeling, both requiring the indication of the regions to be removed or colored. This can be accomplished by using polygonal or curved shapes. For any polygon, we can find the outward normals of its faces. Subproblems include determining whether a line segment and a Gaussian intersect. For a line segment, edge projection can be applied to determine whether an intersection exists. With a Gaussian position μ_0 and a line segment $\{\mathbf{A}, \mathbf{B}\}$,we define:

$$I = ((\mathbf{A} - \mu_0) \cdot (\mathbf{A} - \mathbf{B}))((\mathbf{B} - \mu_0) \cdot (\mathbf{A} - \mathbf{B})) \tag{36}$$

If $I \geq 0$ and the intersection threshold η is guaranteed, the line segment and the Gaussian intersecisFor a more complex polyhedral

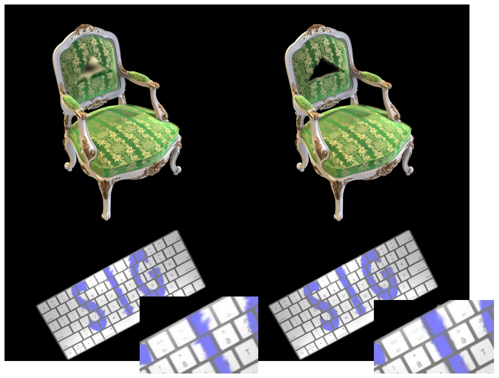

Fig. 7. The examples (top) show the removal of a triangular part from the chair and the examples(bottom) demonstrate mapping a "SIG" curve to a keyboard. EVSplitting produces smooth boundaries in various scenarios while maintaining visual consistency compared to the Filter solution.

Table 5. The E_i and E_e after engraving the chair with different strategies given different shapes.

Engraving Shapes	Engraving Strategy	$E_i \downarrow$	$E_e \downarrow$
polygon (triangle)	Filter solution	0.0000	0.0668
polygon (triangle)	EVSplitting	0.3610	0.0005
curve (circle)	Filter solution	0.0000	0.0710
curve (circle)	EVSplitting	0.3027	0.0015

bounding box, we can use a combination of the above methods or a ray-casting algorithm to find the Gaussians in the bounding box.

For a 3D closed curved surface $B(x, y, z) = 0$, we can compute the tangent normal $\mathbf{n}$ of the closest position on the curve to the Gaussian position as $\mathbf{n} = \frac{\nabla B}{||\nabla B||}$. $\mathbf{n}$ is an outward normal from the inside of the curve according to the definition of the 3D closed curve. With this normal, we can easily determine the influenced Gaussians. We engrave the chair object from the NeRF synthetic dataset [Mildenhall et al. 2020] with different types of shapes. We split the intersected Gaussians with our split algorithm to obtain clearer boundaries. We also compare our splitting algorithm to the "Filter solution", which directly removes or changes the Gaussians' color in the polygon or curve. We report the quantitative results in Table 5, and show the visual comparisons in Figs. 7 and 12. We implemented a pipeline to engrave or texture the 3DGS models from the SVG vector images, which are a series of Bezier curves, and can be obtained from PNG or JPEG images using the Potrace algorithm [Selinger 2003]. More engraving or texturing results using SVG images can be found in Figs. 12 and 13.

4.3 Point Cloud Uniformization

A 3DGS model can be regarded as providing a kind of point cloud, where each point represents a different shape, making the point cloud extracted from 3DGS models non-uniform, with gaps in large flat untextured regions, leading to severe problems for downstream tasks like normal extraction, point cloud segmentation, etc.

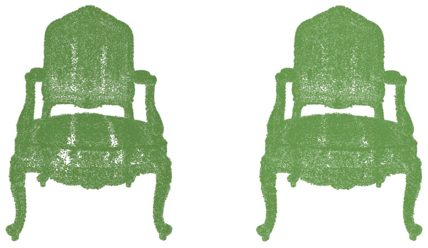

Fig. 8. Our splitting algorithm helps to extract more uniform and denser point clouds from a well-trained 3DGS model for the chair object.

Because EVSplitting can split a big Gaussian to two smaller one, it can be used to increase the uniformity of extracted point clouds. Assume that the average of each Gaussian's maximum principal direction scale is R_m. We can define the degree of inhomogeneity between the i-th largest and the j-th largest eigenvalue ($i < j$) as:

$$\gamma_{ij} = \frac{\mathbf{Tri}[i]}{\mathbf{Tri}[j] + R_m}, (i, j) \in \{(0,1), (1,2), (0,2)\} \tag{37}$$

If any γ_{ij} is greater than the inhomogeneity threshold $\eta_\gamma = 2$, the Gaussian should be split at principal component i. Splitting at the center of a Gaussian will make the positions of new Gaussians too close. To ensure the new Gaussians' positions are far apart, we select the splitting parameter d_0 as $2\mathbf{Tri}[i]$ to ensure a proper interval between new Gaussians. An experimental result is shown in Fig. 8 and more results are shown in Fig. 14. EVSplitting produces a more uniform and denser point cloud than the original 3DGS model and has very few holes.

5 Conclusions and Future Work

In this paper, we model the problem of how to split an N-dimensional Gaussian into two independent N-dimensional Gaussians and present a closed-form solution for this problem. This enables our splitting algorithm to be readily used with any 3DGS model processing. It can be used both to produce to more uniform distributed Gaussians, and to avoid blurring and needle-like artifacts when cutting a Gaussian model by a plane (and indirectly, trimming it in any other way).

In the future, we hope to apply our splitting algorithm not only to other geometric applications of Gaussian splatting models but also to various models that use Gaussian distribution such as VAE [Kingma and Welling 2014] and diffusion [Ho et al. 2020; Song et al. 2021].

Acknowledgments

We thank all the reviewers for their useful suggestions. This work was supported by the National Science and Technology Major Project (2021ZD0112902), the National Natural Science Foundation of China (62220106003), the Research Grant of Beijing Higher Institution Engineering Research Center, the Tsinghua-Tencent Joint Laboratory for Internet Innovation Technology, and the Tsinghua University Initiative Scientific Research Program.

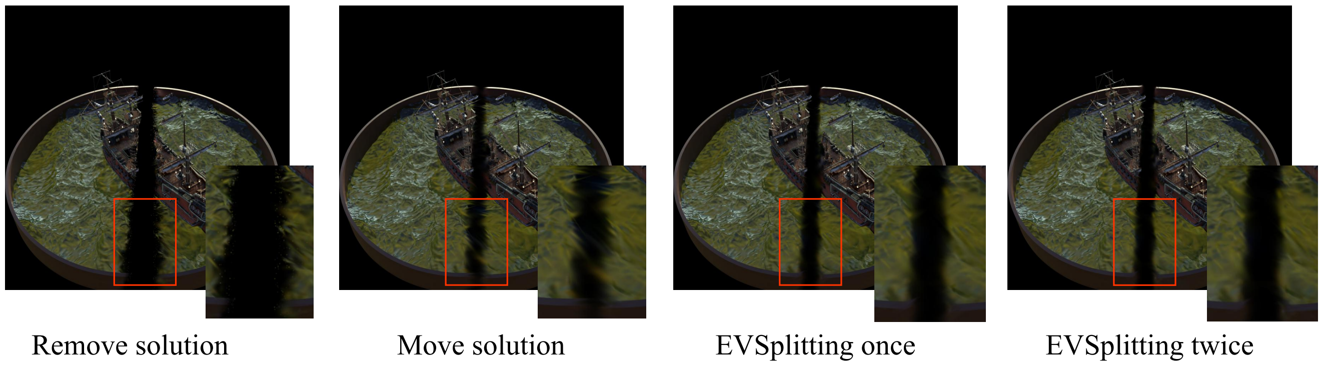

Fig. 9. Plane split results, showing differences between splitting algorithms more clearly. The 'Remove' solution leads to holes, while the 'Move' solution leads to a more uneven boundary. EVSplitting produces a clear boundary between the two parts. Further results are shown in the supplementary video.

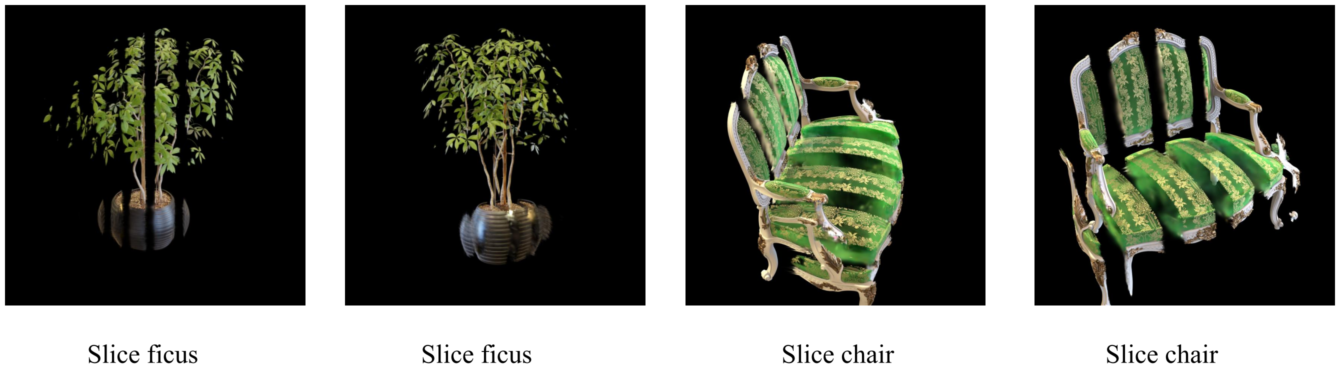

Fig. 10. Slice image on the NeRF synthetic dataset [Mildenhall et al. 2020].

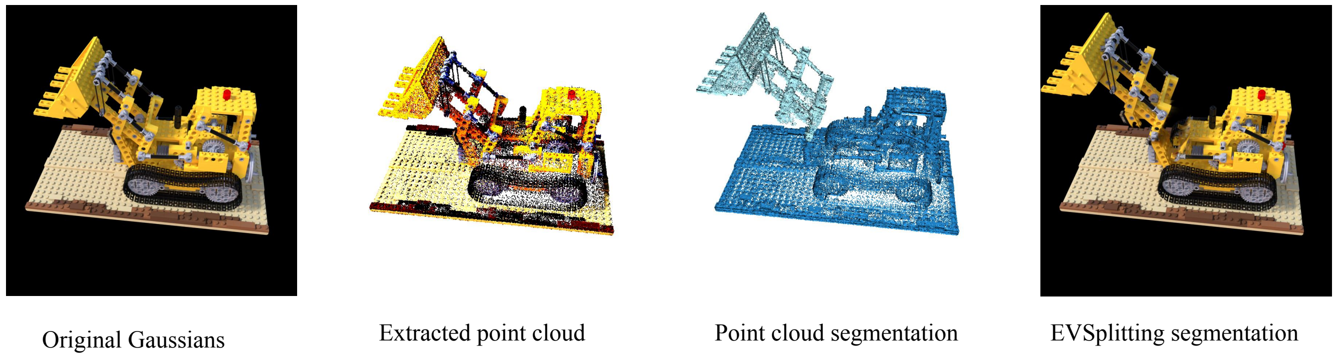

Fig. 11. We applied point cloud segmentation on 3DGS and achieved smooth segmentation boundaries. More results are shown in the supplementary video.

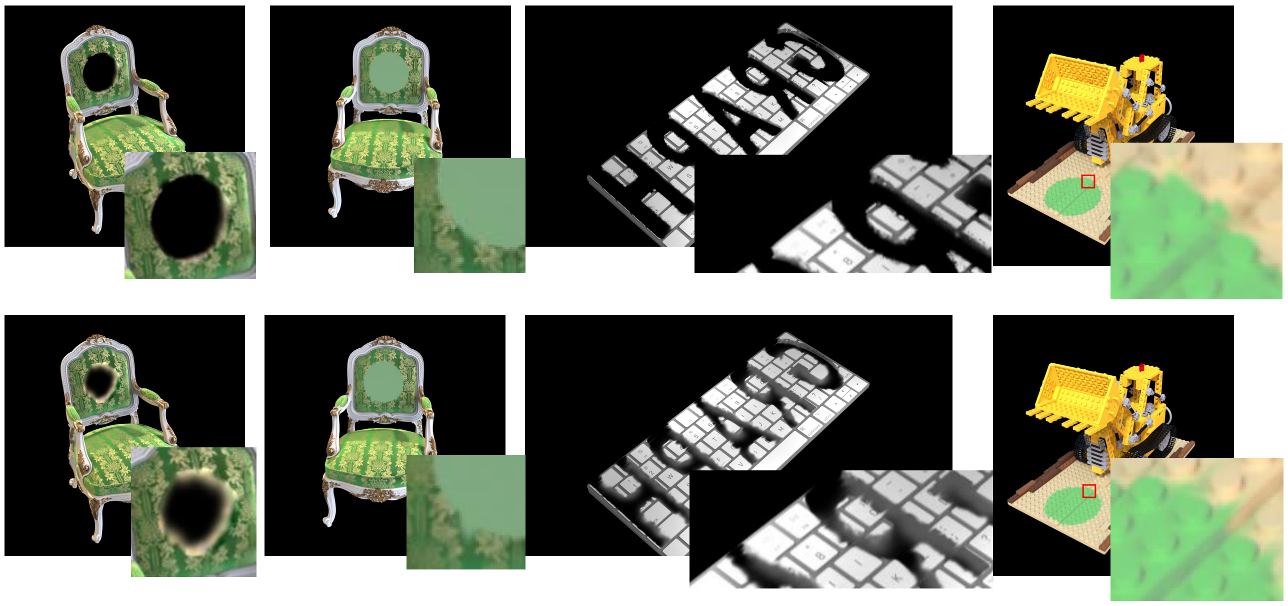

Fig. 12. Visual comparisons between EVSplitting(top) and the "Filter solution" (bottom) on engraving/texturing the chair and keyboard 3DGS models with SVG images. EVSplitting yields clearer boundaries and textures.

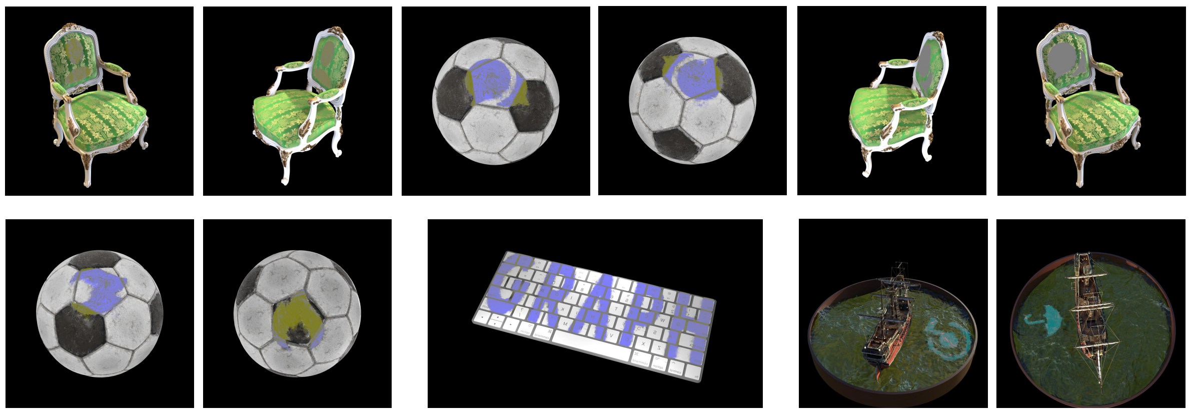

Fig. 13. More results for texturing 3DGS model with SVG images.

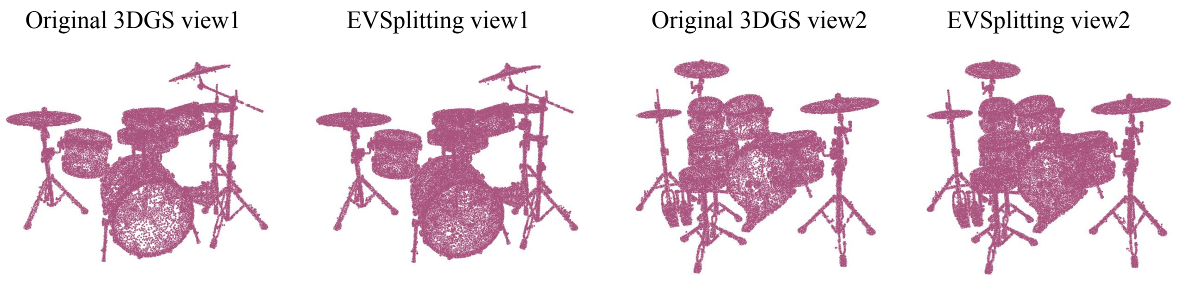

Fig. 14. Example showing that EVSplitting can help extract a more uniform point cloud.

References

Jonathan T. Barron, Ben Mildenhall, Dor Verbin, Pratul P. Srinivasan, and Peter Hedman. 2022. Mip-NeRF 360: Unbounded Anti-Aliased Neural Radiance Fields. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2022, New Orleans, LA, USA, June 18-24, 2022.* IEEE, 5460–5469. https://doi.org/10.1109/CVPR52688.2022.00539

Yiwen Chen, Zilong Chen, Chi Zhang, Feng Wang, Xiaofeng Yang, Yikai Wang, Zhongang Cai, Lei Yang, Huaping Liu, and Guosheng Lin. 2023a. GaussianEditor: Swift and Controllable 3D Editing with Gaussian Splatting. *CoRR* abs/2311.14521 (2023). https://doi.org/10.48550/ARXIV.2311.14521 arXiv:2311.14521

Zilong Chen, Feng Wang, and Huaping Liu. 2023b. Text-to-3D using Gaussian Splatting. *CoRR* abs/2309.16585 (2023). https://doi.org/10.48550/ARXIV.2309.16585 arXiv:2309.16585

Jaeyoung Chung, Suyoung Lee, Hyeongjin Nam, Jaerin Lee, and Kyoung Mu Lee. 2023. Luciddreamer: Domain-free generation of 3d gaussian splatting scenes. *arXiv preprint arXiv:2311.13384* (2023).

Zhi Deng, Haoyao Xiao, Yining Lang, Hao Feng, and Juyong Zhang. 2024. Multi-scale hash encoding based neural geometry representation. *Comput. Vis. Media* 10, 3 (2024), 453–470. https://doi.org/10.1007/S41095-023-0340-X

Jiemin Fang, Junjie Wang, Xiaopeng Zhang, Lingxi Xie, and Qi Tian. 2023. GaussianEditor: Editing 3D Gaussians Delicately with Text Instructions. *CoRR* abs/2311.16037 (2023). https://doi.org/10.48550/ARXIV.2311.16037 arXiv:2311.16037

Meng-Hao Guo, Junxiong Cai, Zheng-Ning Liu, Tai-Jiang Mu, Ralph R. Martin, and Shi-Min Hu. 2021. PCT: Point cloud transformer. *Computational Visual Media* 7, 2 (2021), 187–199.

Yuan-Chen Guo, Yan-Pei Cao, Chen Wang, Yu He, Ying Shan, and Song-Hai Zhang. 2023. VMesh: Hybrid Volume-Mesh Representation for Efficient View Synthesis. In *SIGGRAPH Asia.* ACM, 17:1–17:11.

Peter M. Hall, A. David Marshall, and Ralph R. Martin. 2002. Adding and subtracting eigenspaces with eigenvalue decomposition and singular value decomposition. *Image Vis. Comput.* 20, 13-14 (2002), 1009–1016. https://doi.org/10.1016/S0262-8856(02)00114-2

Michael Himmelsbach, Felix V Hundelshausen, and H-J Wuensche. 2010. Fast segmentation of 3D point clouds for ground vehicles. In *2010 IEEE Intelligent Vehicles Symposium.* IEEE, 560–565.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. 2020. Denoising Diffusion Probabilistic Models. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin (Eds.). https://proceedings.neurips.cc/paper/2020/hash/4c5bcfec8584af0d967f1ab10179ca4b-Abstract.html

Huajian Huang, Longwei Li, Hui Cheng, and Sai-Kit Yeung. 2023. Photo-SLAM: Real-time Simultaneous Localization and Photorealistic Mapping for Monocular, Stereo, and RGB-D Cameras. *CoRR* abs/2311.16728 (2023). https://doi.org/10.48550/ARXIV.2311.16728 arXiv:2311.16728

Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 2023. 3D Gaussian Splatting for Real-Time Radiance Field Rendering. *ACM Trans. Graph.* 42, 4 (2023), 139:1–139:14.

Diederik P. Kingma and Max Welling. 2014. Auto-Encoding Variational Bayes. In *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*, Yoshua Bengio and Yann LeCun (Eds.). http://arxiv.org/abs/1312.6114

Alexander Kirillov, Eric Mintun, Nikhila Ravi, Hanzi Mao, Chloé Rolland, Laura Gustafson, Tete Xiao, Spencer Whitehead, Alexander C. Berg, Wan-Yen Lo, Piotr Dollár, and Ross B. Girshick. 2023. Segment Anything. In *IEEE/CVF International Conference on Computer Vision, ICCV 2023, Paris, France, October 1-6, 2023.* IEEE, 3992–4003. https://doi.org/10.1109/ICCV51070.2023.00371

Yushi Lan, Feitong Tan, Di Qiu, Qiangeng Xu, Kyle Genova, Zeng Huang, Sean Fanello, Rohit Pandey, Thomas Funkhouser, Chen Change Loy, et al. 2023. Gaussian3Diff: 3D Gaussian Diffusion for 3D Full Head Synthesis and Editing. *arXiv preprint arXiv:2312.03763* (2023).

Hidenobu Matsuki, Riku Murai, Paul H. J. Kelly, and Andrew J. Davison. 2023. Gaussian Splatting SLAM. *CoRR* abs/2312.06741 (2023). https://doi.org/10.48550/ARXIV.2312.06741 arXiv:2312.06741

Ben Mildenhall, Pratul P. Srinivasan, Matthew Tancik, Jonathan T. Barron, Ravi Ramamoorthi, and Ren Ng. 2020. NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis. In *Computer Vision - ECCV 2020 - 16th European Conference, Glasgow, UK, August 23-28, 2020, Proceedings, Part I (Lecture Notes in Computer Science, Vol. 12346)*, Andrea Vedaldi, Horst Bischof, Thomas Brox, and Jan-Michael Frahm (Eds.). Springer, 405–421. https://doi.org/10.1007/978-3-030-58452-8_24

Todd K Moon. 1996. The expectation-maximization algorithm. *IEEE Signal processing magazine* 13, 6 (1996), 47–60.

Thomas Müller, Alex Evans, Christoph Schied, and Alexander Keller. 2022. Instant Neural Graphics Primitives with a Multiresolution Hash Encoding. *ACM Trans. Graph.* 41, 4 (2022), 102:1–102:15.

Ben Poole, Ajay Jain, Jonathan T. Barron, and Ben Mildenhall. 2023. DreamFusion: Text-to-3D using 2D Diffusion. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023.* OpenReview.net. https://openreview.net/pdf?id=FjNys5c7VyY

Thomas Porter and Tom Duff. 1984. Compositing digital images. In *Proceedings of the 11th annual conference on Computer graphics and interactive techniques.* 253–259.

Peter Selinger. 2003. Potrace: a polygon-based tracing algorithm.

Tianchang Shen, Jun Gao, Kangxue Yin, Ming-Yu Liu, and Sanja Fidler. 2021. Deep Marching Tetrahedra: a Hybrid Representation for High-Resolution 3D Shape Synthesis. In *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, Marc'Aurelio Ranzato, Alina Beygelzimer, Yann N. Dauphin, Percy Liang, and Jennifer Wortman Vaughan (Eds.). 6087–6101. https://proceedings.neurips.cc/paper/2021/hash/30a237d18c50f563cba4531f1db44acf-Abstract.html

Jiaming Song, Chenlin Meng, and Stefano Ermon. 2021. Denoising Diffusion Implicit Models. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021.* OpenReview.net. https://openreview.net/forum?id=St1giarCHLP

Olga Sorkine and Marc Alexa. 2007. As-rigid-as-possible surface modeling. In *Proceedings of the Fifth Eurographics Symposium on Geometry Processing, Barcelona, Spain, July 4-6, 2007 (ACM International Conference Proceeding Series, Vol. 257)*, Alexander G. Belyaev and Michael Garland (Eds.). Eurographics Association, 109–116. https://doi.org/10.2312/SGP/SGP07/109-116

Jingxiang Sun, Bo Zhang, Ruizhi Shao, Lizhen Wang, Wen Liu, Zhenda Xie, and Yebin Liu. 2024. DreamCraft3D: Hierarchical 3D Generation with Bootstrapped Diffusion Prior. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024.* OpenReview.net. https://openreview.net/forum?id=DDX1u29Gqr

Jiaxiang Tang, Zhaoxi Chen, Xiaokang Chen, Tengfei Wang, Gang Zeng, and Ziwei Liu. 2024a. LGM: Large Multi-View Gaussian Model for High-Resolution 3D Content Creation. *arXiv preprint arXiv:2402.05054* (2024).

Jiaxiang Tang, Jiawei Ren, Hang Zhou, Ziwei Liu, and Gang Zeng. 2024b. DreamGaussian: Generative Gaussian Splatting for Efficient 3D Content Creation. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024.* OpenReview.net. https://openreview.net/forum?id=UyNXMqnN3c

Peng Wang, Lingjie Liu, Yuan Liu, Christian Theobalt, Taku Komura, and Wenping Wang. 2021. NeuS: Learning Neural Implicit Surfaces by Volume Rendering for Multi-view Reconstruction. In *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, Marc'Aurelio Ranzato, Alina Beygelzimer, Yann N. Dauphin, Percy Liang, and Jennifer Wortman Vaughan (Eds.). 27171–27183. https://proceedings.neurips.cc/paper/2021/hash/e41e164f7485ec4a28741a2d0ea41c74-Abstract.html

Yizhi Wang, Wallace Lira, Wenqi Wang, Ali Mahdavi-Amiri, and Hao Zhang. 2023a. Slice3D: Multi-Slice, Occlusion-Revealing, Single View 3D Reconstruction. *arXiv preprint arXiv:2312.02221* (2023).

Zhengyi Wang, Cheng Lu, Yikai Wang, Fan Bao, Chongxuan Li, Hang Su, and Jun Zhu. 2023b. ProlificDreamer: High-Fidelity and Diverse Text-to-3D Generation with Variational Score Distillation. In *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*, Alice Oh, Tristan Naumann, Amir Globerson, Kate Saenko, Moritz Hardt, and Sergey Levine (Eds.). http://papers.nips.cc/paper_files/paper/2023/hash/1a87980b9853e84dfb295855b425c262-Abstract-Conference.html

Guanjun Wu, Taoran Yi, Jiemin Fang, Lingxi Xie, Xiaopeng Zhang, Wei Wei, Wenyu Liu, Qi Tian, and Xinggang Wang. 2023. 4D Gaussian Splatting for Real-Time Dynamic Scene Rendering. *CoRR* abs/2310.08528 (2023). https://doi.org/10.48550/ARXIV.2310.08528 arXiv:2310.08528

Qiangeng Xu, Zexiang Xu, Julien Philip, Sai Bi, Zhixin Shu, Kalyan Sunkavalli, and Ulrich Neumann. 2022. Point-NeRF: Point-based Neural Radiance Fields. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2022, New Orleans, LA, USA, June 18-24, 2022.* IEEE, 5428–5438. https://doi.org/10.1109/CVPR52688.2022.00536

Zeyu Yang, Hongye Yang, Zijie Pan, and Li Zhang. 2024. Real-time Photorealistic Dynamic Scene Representation and Rendering with 4D Gaussian Splatting. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024.* OpenReview.net. https://openreview.net/forum?id=WhgB5sispV

Biao Zhang, Jiapeng Tang, Matthias Nießner, and Peter Wonka. 2023. 3DShape2VecSet: A 3D Shape Representation for Neural Fields and Generative Diffusion Models. *ACM Trans. Graph.* 42, 4 (2023), 92:1–92:16.