BG-Triangle: Bézier Gaussian Triangle for 3D Vectorization and Rendering

Supplementary Material

1. Implementation Details

Code Release We built upon the 3DGS [3] code base to add discontinuity-aware rendering and corresponding acceleration algorithms in the CUDA rendering code. We used PyTorch [4] to implement the proposed rendering pipeline. Our code will be made publicly available to the community once the paper is accepted.

Point Cloud Initialization Our method can be initialized using either a mesh or a coarse point cloud. Generally, point clouds are easier to obtain compared to meshes. A coarse point cloud can be generated from MVS [1] algorithms, 3D scanners, or even 3DGS. In our experiments, for each scene, the point cloud we used was derived from the reconstruction result of the 3000th iteration of 3DGS. We randomly sampled 10,000 points from this result to form the initial point cloud. On each point in the initial point cloud, we placed equilateral triangle primitives with different orientations but the same size as the initialization result.

Cube Initialization BG-Triangle can reconstruct scenes initialized from a cube (Fig. 1). During optimization, BG-Triangles dynamically 'grow' and 'contract' to conform to the shape of the 3D scene.

Optimization with cube initialization requires more iterations to match the reconstruction quality of point-cloud initialization and is more sensitive to hyperparameters (e.g., learning rates). Since point clouds are easily generated from multi-view images during calibration, we use point cloud initialization in our experiments.

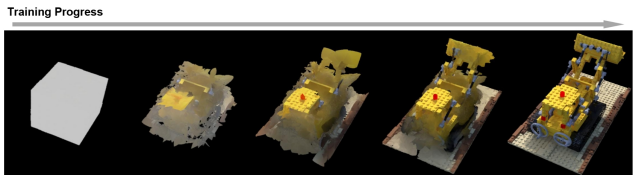

Figure 1. **The visualization of cube initialization** of the training progress on the Lego scene from the NeRF-Synthetic dataset.

Multi-Resolution Attribute maps. In our implementation, the attribute map of each primitive is a multi-channel grid shaped as an isosceles right triangle, with the pixel resolution of the legs being 3, except the SH attribute map whose resolution is 1. The number of channels in the attribute map corresponds to the original dimension of the respective attribute.

Splitting and Pruning. Splitting and pruning are essential for Bézier Gaussian Triangles (BG-Triangles) to effectively represent a scene. The splitting and pruning strategy is performed every 300 iterations and is disabled after the 15,000th iteration to stabilize the optimization process.

For splitting, we follow two pre-defined criteria: position gradient amplitude and edge prior. We compute the position gradient for each BG-Triangle. If the average position gradient of a BG-Triangle exceeds $\tau_g = 0.0018$, the region is considered under-reconstructed. Using the index map $\mathbf{I}_{id}$, we map the edge gradients of the image to the corresponding BG-Triangle and accumulate their values. If this accumulated average exceeds $\tau_b = 13.0$, it indicates that the BG-Triangle may be crossing a boundary. In either case, we subdivide the BG-Triangle into four smaller BG-Triangles by splitting it at the midpoints of its edges.

For pruning, we follow three pre-defined criteria: visibility, area, and aspect ratio of the primitives. Using the index map $\mathbf{I}_{id}$, the visibility count of each BG-Triangle is tracked. When the average visibility count ratio falls below $\tau_v = 0.08$, the BG-Triangle is considered occluded. To further evaluate visibility, we maintain a visibility texture for each BG-Triangle. Using the coordinate map $\mathbf{I}_{uv}$, the UV coordinates are mapped to the visibility texture, and the proportion of visible area is calculated. If the visibility ratio falls below $\tau_r = 0.4$, the BG-Triangle is also considered occluded. Additionally, Bézier triangles with an area smaller than $\tau_a = 3 \times 10^{-4}$ or with an excessively elongated shape (aspect ratio above $\tau_s = 10$) are considered insignificant or noisy. The area is estimated by approximating the BG-Triangle as four planar triangles and summing their areas, while the aspect ratio is determined by computing the aspect ratio of the bounding box of BG-Triangle. In any of these cases, the corresponding BG-Triangles are pruned.

2. Additional Results

Reconstruction Quality We compare the reconstruction quality using Chamfer Distance on the NeRF-Synthetic dataset between our method and 2DGS [2] under its default settings for bounded scene, with the downsampling density set to 0.2. As shown in Table 1, our method achieves comparable results to 2DGS.

Applications The 3D vector representation reconstructed by BG-Triangle from multi-view images shows a clustering effect at the geometric and texture boundaries. In other words, there is a relatively dense distribution of primitives at the boundaries. By leveraging this distribution character-

	Chair	Drums	Ficus	Hotdog
2DGS	**0.0645**	0.0675	0.0667	0.0693
Ours	0.0722	**0.0592**	**0.0535**	**0.0642**
	Lego	Material	Mic	Ship
2DGS	**0.0581**	**0.0540**	0.0629	**0.0735**
Ours	0.0597	0.0561	**0.0628**	0.0786

Table 1. Reconstruction quality comparisons on NeRF-Synthetic Dataset.

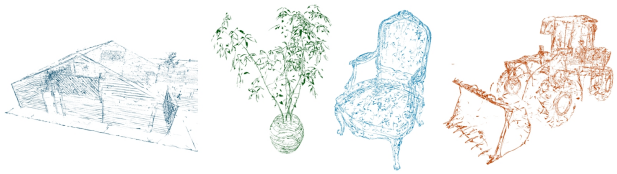

Figure 2. 3D line strokes extracted from dense boundary primitives reveal rich contour information and semantic features of the scene.

istic, we can extract the 3D line strokes at the boundaries by filtering out tessellated primitives based on their face area and removing those with areas that are too large. As shown in the Fig. 2, the 3D line strokes allow us to easily distinguish the content of the scene, indicating that these 3D line strokes contain rich contour information and exhibit certain semantic features.

3. Backward Derivative

We follow the notations defined in Sec. 3.

Derivative of Gradient in Discontinuity-aware Alpha Blending. The alpha value with blending coefficient is:

$$\alpha(\mathbf{q}) = o \cdot w(\mathbf{q}) \cdot \exp\left(-\frac{1}{2}(\mathbf{q}-\mu)^\top \Sigma^{-1}(\mathbf{q}-\mu)\right) \quad (1)$$

where μ is the 2D position of the Gaussian $\mathcal{G}$, Σ is the 2D covariance matrix, o is a constant scalar. In alpha blending, the final color is computed as:

$$C = \sum_{i=0}^{n-1} T_i \alpha_i c_i + T_n c_{\text{bg}} \quad (2)$$

where c_{bg} is the background color, c_i is the color associated with each Gaussian $\mathcal{G}_i$, T_i is the transparent term, defined as:

$$T_i = \prod_{j=0}^{i-1}(1-\alpha_j), \quad T_0 = 1 \quad (3)$$

The gradient of the loss ℓ with respect to color c_i is given by:

$$\frac{\partial \ell}{\partial c_i} = \frac{\partial \ell}{\partial C}\frac{\partial C}{\partial c_i} = \frac{\partial \ell}{\partial C} T_i \alpha_i. \quad (4)$$

The gradient of ℓ with respect to alpha α_i is:

$$\begin{aligned}\frac{\partial \ell}{\partial \alpha_i} &= \frac{\partial \ell}{\partial C}\frac{\partial C}{\partial \alpha_i} \\ &= \frac{\partial \ell}{\partial C}\left[\left(\sum_{j=0}^{n-1}\frac{\partial T_j}{\partial \alpha_i}\alpha_j c_j + T_j \frac{\partial \alpha_j}{\partial \alpha_i} c_j\right) + \frac{\partial T_n}{\partial \alpha_i} c_{\text{bg}}\right] \\ &= \frac{\partial \ell}{\partial C}\left[T_i c_i - \frac{1}{1-\alpha_i}\left(\sum_{j=i+1}^{n-1} T_j \alpha_j c_j + T_n c_{\text{bg}}\right)\right].\end{aligned} \quad (5)$$

The gradient of ℓ with respect to w is:

$$\frac{\partial \ell}{\partial w} = \frac{\partial \ell}{\partial \alpha}\frac{\partial \alpha}{\partial w} = \frac{\partial \ell}{\partial \alpha}\frac{\alpha}{w}. \quad (6)$$

Boundary pixels, denoted as $\mathcal{B}$, are derived from the index map $\mathbf{I}_{\text{id}}$, which is ultimately determined by the control points of Bézier triangles. To account for these boundary pixels, we calculate the gradient of ℓ with respect to a boundary pixel $\mathbf{b} \in \mathcal{B}$:

$$\begin{aligned}\frac{\partial \ell}{\partial \mathbf{b}} &= \frac{\partial \ell}{\partial w}\frac{\partial w}{\partial \mathbf{b}} \\ &= \begin{cases} 0 & \text{if } \mathbf{I}_{id}(\mathbf{b}) <> e, \\ \dfrac{\partial \ell}{\partial w}\gamma'(\|\mathbf{q}-\mathbf{b}\|;\sigma)\dfrac{\mathbf{q}-\mathbf{b}}{\|\mathbf{q}-\mathbf{b}\|} & \text{if } \mathbf{I}_{id}(\mathbf{b}) = \mathbf{I}_{id}(\mathbf{q}) \\ -\dfrac{\partial \ell}{\partial w}\gamma'(\|\mathbf{q}-\mathbf{b}\|;\sigma)\dfrac{\mathbf{q}-\mathbf{b}}{\|\mathbf{q}-\mathbf{b}\|} & \text{otherwise} \end{cases}\end{aligned} \quad (7)$$

where

$$\begin{aligned}\gamma'(\|\mathbf{q}-\mathbf{b}\|;\sigma) &= \frac{\ln 2}{\sigma} 2^{\frac{\|\mathbf{q}-\mathbf{b}\|}{\sigma}-1} \\ &= \frac{\ln 2}{\sigma}\gamma(\|\mathbf{q}-\mathbf{b}\|;\sigma).\end{aligned} \quad (8)$$

Other factors related to the attributes of the Gaussian $\mathcal{G}$ can be handled automatically by the 3DGS framework, and are therefore omitted here for brevity.

Derivative of Gradient in Sub-primitive Generation. Both the 3D position $\mathbf{S}_\mathbf{q}$ and the diffuse color $\mathbf{c}_\mathbf{q}$ of the Gaussian $\mathcal{G}$ can be interpolated barycentrically by the control points. Taking the 3D position $\mathbf{S}_\mathbf{q}$ as an example, the gradient of the loss ℓ with respect to a control point $\mathbf{p}_{i,j,k}(\mathbf{I}_{id}(\mathbf{q}))$ is given by:

$$\begin{aligned}\frac{\partial \ell}{\partial \mathbf{p}_{i,j,k}(\mathbf{I}_{id}(\mathbf{q}))} &= \frac{\partial \ell}{\partial \mathbf{S}_\mathbf{q}}\frac{\partial \mathbf{S}_\mathbf{q}}{\partial \mathbf{p}_{i,j,k}(\mathbf{I}_{id}(\mathbf{q}))} \\ &= \frac{\partial \ell}{\partial \mathbf{S}_\mathbf{q}} B^n_{i,j,k}(\mathbf{I}_{uv}(\mathbf{q})).\end{aligned} \quad (9)$$

where $B^n_{i,j,k}$ is the Bernstein polynomial of degree n defined on the barycentric coordinate system.

Figure 3. **Additional results of our BG-Triangle.**

4. Result Gallery

We provide additional results of our BG-Triangle in Fig. 3.

5. Comparison Setup

We first set an upper control on the number of primitives for different methods based on a predefined parameter scale. Starting with sparse point clouds, we optimize and proceed to the splitting phase. During this phase, we check the total number of primitives after splitting. If the total number remains within the control, we keep the splitting results. If it exceeds the control, we randomly select newly added primitives, ensuring the total number adheres to the control.

In addition, pruning operations for each method remain enabled to dynamically adjust the primitive distribution effectively. When pruning, we disable pruning large Gaussian primitives to ensure that different methods can fit the overall structure of the object. This approach balances dynamic adjustments with fitting precision.

6. Limitations

Similar to the vectorization process for 2D images, which often entails a loss of image quality, the representations reconstructed and rendered using our method exhibit a quality that remains slightly below that achieved by state-of-the-art novel view synthesis techniques. At the same time, our approach, which generates only a single layer of sub-primitives, cannot handle materials with opacity.

References

[1] Michael Goesele, Noah Snavely, Brian Curless, Hugues Hoppe, and Steven M. Seitz. Multi-view stereo for community photo collections. In *2007 IEEE 11th International Conference on Computer Vision*, pages 1–8, 2007. 1

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

[3] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 3d gaussian splatting for real-time radiance field rendering. *ACM Transactions on Graphics*, 42(4), 2023. 1

[4] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32, 2019. 1