3D Convex Splatting: Radiance Field Rendering with 3D Smooth Convexes

Jan Held[* 1,2]   Renaud Vandeghen[* 1]   Abdullah Hamdi[* 3]   Adrien Deliege[1]   Anthony Cioppa[1]
Silvio Giancola[2]   Andrea Vedaldi[3]   Bernard Ghanem[2]   Marc Van Droogenbroeck[1]

[1] University of Liège   [2] KAUST   [3] University of Oxford

∗ Equal contribution

Abstract

Recent advances in radiance field reconstruction, such as 3D Gaussian Splatting (3DGS), have achieved high-quality novel view synthesis and fast rendering by representing scenes with compositions of Gaussian primitives. However, 3D Gaussians present several limitations for scene reconstruction. Accurately capturing hard edges is challenging without significantly increasing the number of Gaussians, creating a large memory footprint. Moreover, they struggle to represent flat surfaces, as they are diffused in space. Without hand-crafted regularizers, they tend to disperse irregularly around the actual surface. To circumvent these issues, we introduce a novel method, named 3D Convex Splatting (3DCS), which leverages 3D smooth convexes as primitives for modeling geometrically-meaningful radiance fields from multi-view images. Smooth convex shapes offer greater flexibility than Gaussians, allowing for a better representation of 3D scenes with hard edges and dense volumes using fewer primitives. Powered by our efficient CUDA-based rasterizer, 3DCS achieves superior performance over 3DGS on benchmarks such as Mip-NeRF360, Tanks and Temples, and Deep Blending. Specifically, our method attains an improvement of up to 0.81 in PSNR and 0.026 in LPIPS compared to 3DGS while maintaining high rendering speeds and reducing the number of required primitives. Our results highlight the potential of 3D Convex Splatting to become the new standard for high-quality scene reconstruction and novel view synthesis. The project page is https://convexsplatting.github.io

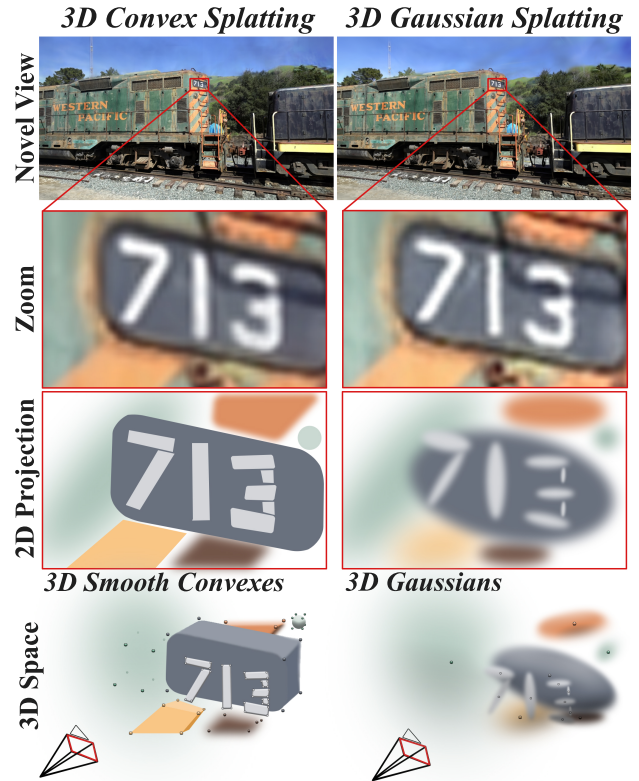*3D Convex Splatting*   *3D Gaussian Splatting*

Figure 1. **3D Convex Splatting for Novel View Synthesis.** We introduce a novel primitive-based pipeline for novel view synthesis with 3D smooth convexes. Our 3D smooth convexes share the rendering speed of 3D Gaussians [26] and the flexible representation of smooth convexes [9]. As a result, 3D Convex Splatting better reconstructs scenes with fewer primitives.

1. Introduction

Reconstructing complex scenes and synthesizing novel views have been fundamental challenges in computer vision and graphics for decades [1, 12], with applications ranging from virtual reality to autonomous navigation [17, 36, 40]. Neural Radiance Fields (NeRF) [38] revolutionized this area by representing scenes as continuous volumetric radi-ance fields, which are optimized to render novel views at high-quality. However, NeRF suffered from slow training and rendering times, limiting its practicality. To address these issues, 3D Gaussian Splatting (3DGS) [26] emerged as an efficient alternative, by representing scenes with millions of 3D Gaussian primitives. 3DGS significantly accelerated training and enabled real-time rendering while main-

taining high-quality outputs.

Despite this progress, Gaussian primitives have two inherent limitations. (1) Gaussians lack defined physical boundaries, making them unsuitable for accurately representing flat surfaces or enabling physically meaningful decompositions of scenes. (2) In addition to their specific smoothness and rounded nature, Gaussians are inadequate for capturing hard edges and geometric structures. Each Gaussian behaves similarly to an ellipsoid, with a symmetrical distribution, struggling to conform to angular boundaries or flat surfaces. This inherent limitation is highlighted by the sphere packing problem in computer graphics [8, 16], where densely packed spherical or ellipsoidal shapes leave gaps and result in inefficient coverage, especially along flat or sharp corners. As with spheres or ellipsoids, an impractically large number of Gaussian primitives would be needed to fill space without gaps, leading to increased memory consumption and computational overhead.

To overcome these limitations, we propose a novel method called *3D Convex Splatting (3DCS)*, which leverages *3D smooth convexes* as primitives for modeling and reconstructing geometrically accurate radiance fields from multi-view images. 3D smooth convexes offer greater flexibility than Gaussians, as they can form dense volumes that accurately capture hard edges and detailed surfaces using fewer primitives. Figure 1 illustrates this point, showing that 3DCS enables the rendering of 3D smooth convexes to generate high-quality novel views of complex scenes. Moreover, by incorporating *smoothness* and *sharpness* parameters, we can control the curvature and the diffusion of the smooth convexes, respectively. This enables the creation of shapes that are hard or soft, dense or diffuse. Figure 2 shows a toy example of how smooth convexes can represent a chair with hard edges with far fewer elements than Gaussians, while utilizing the same optimization. For novel view synthesis, we merge the benefits of the fast rendering process from Gaussians [26], with the flexibility of 3D smooth convexes [9]. We achieve this by rendering 3D smooth convexes using our efficient CUDA-based rasterizer, which enables real-time rendering and accelerates the optimization process. To the best of our knowledge, 3D Convex Splatting is the first method to leverage differentiable smooth convex shapes for novel view synthesis on realistic scenes, outperforming previous methods that use other primitives.

Contributions. We summarize our contributions as follows: **(i)** We introduce 3D Convex Splatting (3DCS), utilizing 3D smooth convexes as novel primitives for radiance field representation, addressing the limitations of Gaussian primitives in capturing dense volumetric features. **(ii)** We develop an optimization framework and a fast, differentiable GPU-based rendering pipeline for our 3D smooth convexes, enabling high-quality 3D scene representations from multi-view images and high rendering speeds.
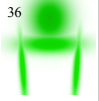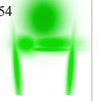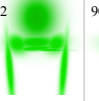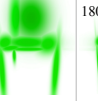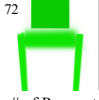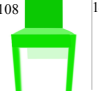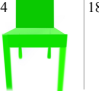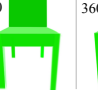

Figure 2. **Toy Experiment of Modeling a Chair.** For the chair input image, we use Gaussians and smooth 6-point convexes to fit the chair with an increasing number of primitives. Note how the convexes efficiently represent the chair with fewer parameters.

(iii) 3DCS surpasses existing rendering primitives on Mip-NeRF360, Tanks and Temples, and Deep Blending datasets, achieving better performance than 3D Gaussian Splatting while using a reduced number of primitives per scene.

2. Related Work

Neural radiance fields (NeRF). Recovering the 3D structure of a scene from images captured from multiple viewpoints is a fundamental problem in computer vision [1, 12]. The introduction of Neural Radiance Fields (NeRF) [38] revolutionized this field by representing scenes as volumetric radiance fields, enabling high-quality novel view synthesis [2, 3, 50]. NeRF employs multi-layer perceptrons to encode scene geometry and view-dependent appearance, optimized via photometric loss through volume rendering [10, 24, 30, 37]. Enhancements to NeRF include grid-based representations for faster training [6, 13, 39, 48], baking techniques for accelerated rendering [20, 45, 46, 52], as well as addressing challenges such as antialiasing [3, 4], modeling unbounded scenes [3, 55], and adapting to few-shot [11, 23, 27] and one-shot settings [5, 53]. In this work, we do not rely on neural networks to model radiance fields like other NeRFs, but instead optimize 3D smooth convexes to fit 3D scenes efficiently. Yet, 3D Convex Splatting allows for strong modeling capacity that rivals MipNeRF-360 [3] in visual fidelity but with real-time rendering speed.

Primitive-based differentiable rendering. Differentiable rendering techniques enable gradient computation through the rendering pipeline, facilitating the optimization of scene parameters from image observations [15, 25, 31–33, 43]. Neural point-based rendering [25] represents scenes with points that store learned features for geometry and texture. 3D Gaussian Splatting (3DGS) [26] introduces Gaussian primitives parameterized by positions, covariances, and appearance attributes. By optimizing millions of Gaussians, 3DGS and achieves high-quality rendering with significantly faster training times and real-time rendering ca-

Inherits Smoothness δ and Sharpness σ

3D Smooth Convex   3D Point Set   2D Projection   Convex Hull and Lines   Signed Distances $\Phi(x,\delta)$ Indicator Function
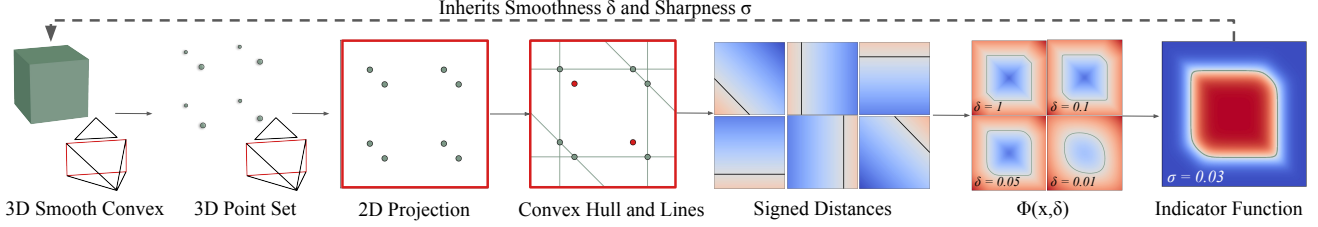
Figure 3. **Convex Splatting Pipeline.** The 3D smooth convex is represented with a point set that is projected in the 2D camera plane. We extract the line-delimited convex hull of the projected points and define the signed distance function for each line. The lines are combined to define an indicator function for each pixel based on smoothness δ and sharpness σ of the 3D convex. The pipeline is differentiable end-to-end, which allows the parameters of the smooth convex primitives to be optimized based on the rendered images.

pabilities. Enhancements to this approach include antialiasing techniques [54], exact volumetric rendering of ellipsoids [35], and extensions to dynamic scene modeling [34]. However, Gaussian primitives have inherent limitations due to their very specific smoothness, making it challenging to capture hard edges and dense volumetric structures without significantly increasing the number of primitives. This leads to increased memory consumption and computational overhead, hindering scalability and efficiency. Alternative primitives [22] have been explored to improve geometric representation. GES [18] utilizes generalized exponential functions to better capture signals with harder edges. 2D Gaussian splatting [21] collapses the 3D Gaussians into oriented planar Gaussians to better represent surfaces. In our work, we introduce 3D smooth convex shapes as a novel primitive for real-time rendering of novel views. These shapes address the limitations of the Gaussian primitives by efficiently capturing dense volumetric shapes.

Convex shapes. Convex shapes have been extensively studied in computer graphics and computer vision due to their geometric simplicity and flexibility in representing complex objects [14, 44]. In 3D reconstruction, convex shape representations have proven highly effective for decomposing complex structures into simpler components like planes, spheres, cubes, cylinders, and superquadrics [42, 49, 51]. CvxNet [9] and BSP-Net [7] introduce neural networks that learn hyperplanes to construct flexible convex shapes, enabling more accurate differentiable modeling of geometries with primitives. A concurrent work utilizes rigid convex polyhedra and differentiable mesh rendering to fit simple 3D shapes with few primitives using multi-view supervision [47]. However, these methods are limited to simple shapes with few optimized primitives and do not scale to large scenes or allow for accurate novel view synthesis.

In our work, we introduce splatting-based rasterization with an array of smooth, high-capacity primitives. This allows us to achieve rendering fidelity levels comparable to volume rendering techniques in modeling complex scenes.

3. Methodology

3D Convex Splatting (3DCS) combines smooth convexes (Sec. 3.1) from CvxNet [9] with the primitive-based representation from 3DGS [26] for efficient, real-time novel view synthesis. 3DCS uses a point-based convex shape representation and convex hull operations to enable straightforward differentiable projection of 3D convex shapes onto the 2D image plane (Sec. 3.2). Further operations for smoothing, splatting, and adaptive densification are used to optimize the representation from posed images (Sec. 3.3). Figure 3 shows an overview of our convex splatting pipeline.

3.1. Preliminaries on 3D Smooth Convexes

Following CvxNet [9], we define a convex polyhedron with J planes ($J > 3$). We define the signed distance $L_j(\mathbf{p})$ between a point $\mathbf{p} \in \mathbb{R}^3$ and a plane $\mathcal{H}_j$ as follows:

$$L_j(\mathbf{p}) = \mathbf{n}_j \cdot \mathbf{p} + d_j \qquad (1)$$

with $\mathbf{n}_j$ being the plane normal pointing towards the outside of the shape and d_j its offset. The signed distance from $\mathbf{p}$ to the convex shape is calculated as the maximum of all the J distances defined by $\tilde{\phi}(\mathbf{p}) = \max_{j=1,\cdots,J} L_j(\mathbf{p})$. However, to create a smooth representation of the convex shape, we use a smooth approximate signed distance function $\phi(\mathbf{p}) \approx \tilde{\phi}(\mathbf{p})$ using the LogSumExp function from CvxNet [9]:

$$\phi(\mathbf{p}) = \log\left(\sum_{j=1}^{J} \exp\left(\delta\, L_j(\mathbf{p})\right)\right), \qquad (2)$$

where the *smoothness* parameter $\delta > 0$ controls the curvature of the convex approximation. Larger values of δ approximate $\phi(\mathbf{p})$ to $\tilde{\phi}(\mathbf{p})$ more closely, resulting in harder edges, while smaller values soften the vertices.

The indicator function $I(\mathbf{p})$ of the smooth convex is then defined by applying a sigmoid function to the approximate signed distance function [9]:

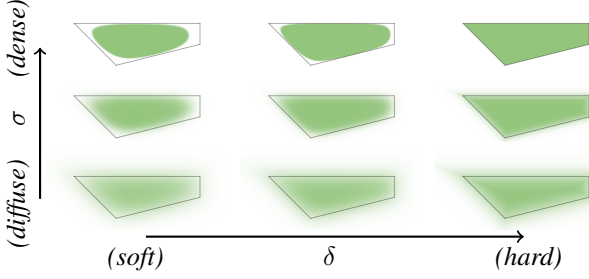$$I(\mathbf{p}) = \text{Sigmoid}\left(-\sigma\, \phi(\mathbf{p})\right), \qquad (3)$$

Figure 4. **Effects of δ and σ on Splatting.** The smoothness δ characterizes vertices and edges, from soft to hard, while the sharpness σ characterizes radiance field transitions, from diffuse to dense.

where the *sharpness* parameter $\sigma > 0$ controls how rapidly the indicator function transitions at the boundary of the underlying convex shape. Higher values of σ result in a steeper transition, making the shape's boundary more defined, whereas lower values result in a more diffuse shape. At the boundary of the convex shape, where $\phi(\mathbf{p}) = 0$, the indicator function of the smooth convex satisfies $I(\mathbf{p}) = 0.5$. More details about smooth convexes can be found in [9]. Fig. 4 illustrates the effect of sharpness σ and smoothness δ on the indicator function $I(.)$.

3.2. 3DCS: Splatting 3D Smooth Convexes

Point-based 3D convex shape representation. Plane-based representations of 3D convexes are impractical for camera plane projections. Unlike CvxNet [9], we define a 3D convex as the convex hull of a 3D point set $\mathbb{S} = \{\mathbf{p}_1, \mathbf{p}_2, \ldots, \mathbf{p}_K\}$. During optimization, the 3D points can move freely, allowing for flexible positioning and morphing of the convex shape. Note that this set of K points does not necessarily correspond to the *explicit* vertices of a convex polyhedron, but rather the hull of the 3D convex shape.

Differentiable projection onto the 2D image plane. To be efficient, we do not explicitly build the 3D convex hull and project it into 2D, but instead, we project the 3D points into 2D and then construct its 2D convex hull. Specifically, we project each 3D point $\mathbf{p}_k \in \mathbb{S}$ onto the 2D image plane using the pinhole perspective camera projection model. The projection involves the intrinsic camera matrix $\mathbf{K}$ and extrinsic parameters (rotation $\mathbf{R}$ and translation $\mathbf{t}$):

$$\mathbf{q}_k = \mathbf{K}\left(\mathbf{R}\mathbf{p}_k + \mathbf{t}\right), \quad \forall k = 1, 2, \ldots, K. \quad (4)$$

This projection is differentiable, allowing gradients to flow back to the 3D points during optimization.

2D convex hull computation. To construct the convex shape in 2D, we apply the Graham Scan algorithm [14], which efficiently computes the convex hull by retaining only the points that define the outer boundary of the projected shape. This approach ensures that the 2D projection

accurately represents the convex outline needed for rendering. The Graham Scan starts by sorting the points based on their polar angle relative to a reference point. After sorting, the Graham Scan algorithm is applied to construct the convex hull by iteratively adding points to the hull while maintaining convexity. The convexity is ensured by checking the cross product of the last two points $\mathbf{q}_i$ and $\mathbf{q}_j$ on the convex hull and the current point $\mathbf{q}_k$, removing points that form a right turn (negative cross product).

Differentiable 2D convex indicator function. We define the 2D convex indicator function of our convex hull by extending the smooth convex representation from 3D to 2D, reusing the equations introduced in Sec. 3.1. We define $\phi(\mathbf{q})$ and $I(\mathbf{q})$ as in Eqs. (2) and (3), but replace the 3D point $\mathbf{p}$ with the 2D point $\mathbf{q}$ and the planes delimiting the 3D convex hull by the lines delimiting the resulting 2D convex hull. The parameters σ and δ are inherited from the 3D smooth convex (from Eqs. (2) and (3)) and still control the sharpness and smoothness of the projected 2D shape boundaries. To account for perspective effects in the 2D projection, we scale δ and σ by the distance d, ensuring that the appearance of the convex shape remains consistent with respect to its distance to the camera.

Efficient differentiable rasterizer. To enable real-time rendering, we build our rasterizer following the 3DGS tile-based rasterizer [26], which allows for efficient backpropagation across an arbitrary number of primitives. All computations, including 3D-to-2D point projection, convex hull calculation, line segment definition, and indicator function implementation—are fully differentiable and executed within our custom CUDA kernels to maximize efficiency and rendering speed. During rendering, we rasterize the 3D shape into the target view using α-blending. For a given camera pose θ and N smooth convexes to render each pixel $\mathbf{q}$, we order the N convexes by sorting them according to increasing distance defined from the camera to their centers, and compute the color value for each pixel $\mathbf{q}$:

$$C(\mathbf{q}) = \sum_{n=1}^{N} \mathbf{c}_n o_n I(\mathbf{q}) \left(\prod_{i=1}^{n-1} \left(1 - o_i I(\mathbf{q})\right) \right), \quad (5)$$

where $\mathbf{c}_n$ is the color of the n-th smooth convex, stored as spherical harmonics and converted to color based on the pose θ, o_n the opacity of the n-th shape, and $I(\mathbf{q})$ is the indicator function adapted to our case from Eq. (3).

3.3. Optimization

Initialization and losses. We optimize the position of each point set in 3D, δ and σ parameters, the opacity o, and the spherical harmonic color coefficients $\mathbf{c}$. To constrain opacity within the range $[0, 1]$, we apply a sigmoid activation function. For δ and σ, we use an exponential activation to
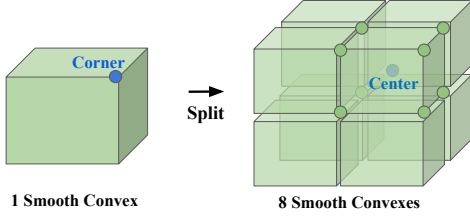
Figure 5. **Adaptive Convex Densification Scheme.** We divide each convex, here exemplified with $K = 8$ points, into as many scaled-down occurrences of the convex, centered at the initial points, each with reduced opacity.
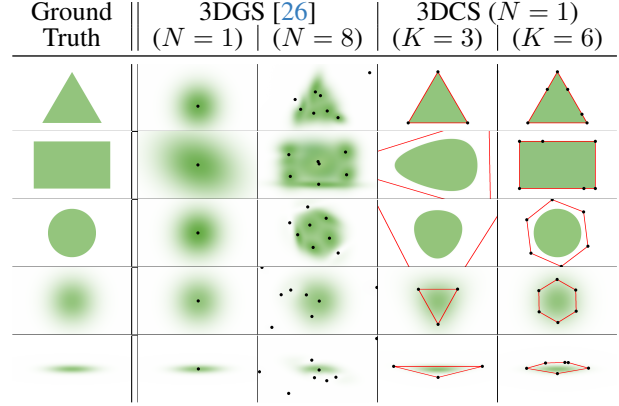

Figure 6. **Reconstruction of Simple Shapes with Primitives.** Smooth convex primitives reconstruct simple shapes better than Gaussians, as they can create sharper geometric boundaries. For 3DCS, the red lines describe the convex hull, whereas the black dots represent the point set. For 3DGS, the black dots represent the Gaussian centers.

ensure their values remain positive. We initialize each convex shape with a set of points uniformly distributed around a sphere centered in the points of the point cloud, using the Fibonacci sphere algorithm. We define the size of each convex shape based on its average distance to the three nearest smooth convexes. This results in smaller smooth convexes in densely populated regions and larger shapes in sparser areas, allowing for adaptive scaling based on local geometry. Following the approach used in 3DGS, we apply a standard exponential decay scheduling technique for the learning rate similar to Plenoxels [13], but only for optimizing the position of the 3D points. We experimented with applying this technique to δ and σ as well, but we did not observe any performance improvements. During training, we use the same regularization loss $\mathcal{L}_m$ as in [29] to reduce the number of convexes. Our final loss function combines $\mathcal{L}_1$ with a D-SSIM term and the loss for the mask, following [29]:

$$\mathcal{L} = (1 - \lambda)\mathcal{L}_1 + \lambda\mathcal{L}_{\text{D-SSIM}} + \beta\mathcal{L}_m , \qquad (6)$$

where λ controls the balance between $\mathcal{L}_1$ and $\mathcal{L}_{\text{D-SSIM}}$. For all tests, we use $\lambda = 0.2$ as in 3DGS [26] and $\beta = 0.0005$.

Adaptive convex shape refinement. The initial set of smooth convexes is generated from a sparse point set obtained through Structure-from-Motion. Since this initial number of smooth convexes is insufficient to accurately represent complex scenes, we employ an adaptive control mechanism to add smooth convexes dynamically. In 3DGS, additional Gaussians are introduced by splitting or cloning those with large view-space positional gradients. However, in 3DCS, positional gradients do not consistently correspond to regions with missing geometric features ("under-reconstruction") or areas where convexes over-represent large portions of the scene ("over-reconstruction"). Instead, we observe that 3DCS exhibits a large sharpness σ loss in both under-reconstructed and over-reconstructed regions. Rather than cloning and differentiating between small and large shapes, we consistently split our smooth convexes. Instead of splitting a smooth convex into just two new convexes, we split it directly into K new convexes. Each new convex shape is scaled down, and the centers of these new

convexes correspond to the K points defining the initial convex shape. By placing the centers of the new convexes at the 3D points of the initial convex shape, we ensure that the new shapes collectively cover the volume of the original convex to maintain the overall completeness of the 3D representation (see Fig. 5 for illustration). To encourage the formation of denser volumetric shapes during optimization, we increase the sharpness σ throughout splitting, while keeping the smoothness δ the same. We prune convexes that have an opacity lower than a predefined threshold and convexes that are too large. More details are provided in the experimental setup Sec. 4.2.

4. Experiments

We first evaluate 3D Convex Splatting (3DCS) with synthetic experiments to showcase its superior shape representation over Gaussian primitives. Then, we describe the real-world setup, including tasks, datasets, baselines, metrics, and implementation, followed by results on novel view synthesis and an ablation study.

4.1. Experiments on Synthetic Data

Figure 6 compares the representation capabilities of using 1 or 8 Gaussian primitives against using a single convex shape defined by 3 or 6 points. The results demonstrate that smooth convexes effectively approximate a wide range of shapes, including both polyhedra and Gaussians, while requiring fewer primitives for accurate representation.

4.2. Experimental Setup

Datasets. To evaluate 3DCS on real-world novel view synthesis, we use the same datasets as 3DGS [26]. This

Dataset	Tanks&Temples						Deep Blending						Mip-NeRF360 Dataset					
Method—Metric	$LPIPS^{\downarrow}$	$PSNR^{\uparrow}$	$SSIM^{\uparrow}$	Train↓	FPS↑	Mem↓	$LPIPS^{\downarrow}$	$PSNR^{\uparrow}$	$SSIM^{\uparrow}$	Train↓	FPS↑	Mem↓	$LPIPS^{\downarrow}$	$PSNR^{\uparrow}$	$SSIM^{\uparrow}$	Train↓	FPS↑	Mem↓
Instant-NGP[39]	0.305	21.92	0.745	7m	14.4	48MB	0.390	24.96	0.817	8m	2.79	48MB	0.331	25.59	0.699	7.5m	9.43	48MB
ZipNeRF[4]	-	-	-	-	-	-	-	-	-	-	-	-	0.189	28.54	0.828	5h	0.184	569MB
Mip-NeRF360[3]	0.257	22.22	0.759	48h	0.14	8.6MB	0.245	29.40	0.901	48h	0.09	8.6MB	0.237	27.69	0.792	48h	0.06	8.6MB
3DGS[26]	0.183	23.14	0.841	26m	154	411MB	0.243	29.41	0.903	36m	137	676MB	0.214	27.21	0.815	42m	134	734MB
GES [18]	0.198	23.35	0.836	21m	210	222MB	0.252	29.68	0.901	30m	160	399MB	0.250	26.91	0.794	32m	186	377MB
2DGS[21]†	0.212	23.13	0.831	14m	122	200MB	0.257	29.50	0.902	28m	76	353MB	0.252	27.18	0.808	29m	64	484MB
3DCS (light)	0.170	23.71	0.842	46m	40	83 MB	0.245	29.61	0.901	84m	30	110 MB	0.266	26.66	0.769	53m	47	77MB
3DCS	0.157	23.95	0.851	60m	33	282MB	0.237	29.81	0.902	71m	30	332 MB	0.207	27.29	0.802	87m	25	666MB

Table 1. **Comparative Analysis of Novel View Synthesis Methods.** We conduct a quantitative comparison of our 3DCS method across three datasets. 3DCS achieves higher-quality results in novel view synthesis with reduced memory consumption, all while achieving fast rendering performance. No codebooks or post-processing compression [29, 41] are used to reduce the size of any of the methods. The best performances are shown in red and the second best in orange. † indicates reproduced results.

includes two scenes from Deep Blending (DB) [19], two scenes from Tanks and Temples (T&T) [28], and all scenes from the Mip-NeRF360 dataset [3].

Baselines. We compare our 3D Convex Splatting method against methods that use three other basic primitives for novel-view synthesis: 3D Gaussians [26], Generalized Exponential Functions (GES) [18], and 2D Gaussians [21]. While many follow-up studies have built upon 3DGS and introduced various enhancements, we focus on the basic primitives for comparison. This choice ensures that the evaluation is based on the core principles of each approach. We compare our method to ZipNeRF [4] and Mip-NeRF360[3] for quality and Instant-NGP [39] for speed.

Metrics. We assess visual quality using SSIM, PSNR, and LPIPS and report training time, rendering speed, and memory usage for a detailed comparison with other methods.

Implementation details of 3DCS. For each experiment, we initialize the number of points per convex shape to $K = 6$ and a spherical harmonic degree of 3, resulting in a total of 69 parameters per convex shape. For each 3D Gaussian, 59 parameters are needed. In the following sections, we compare two variants of 3DCS: a best-performing model and a lightweight variant. Our best model employs different hyperparameters for indoor and outdoor scenes, whereas the lightweight model uses a unified set of parameters. For our light version, we increase the threshold criterion for densifying convexes, effectively reducing the number of shapes. Furthermore, we store the 3D convex parameters with 32-bit precision for the high-quality model and 16-bit precision for the lightweight version. A list of hyperparameters can be found in the *Supplementary Material*. Notably, no compression methods are applied to reduce memory usage.

4.3. Real-world Novel View Synthesis

Main results. Table 1 presents the quantitative results. As can be seen, our 3DCS method consistently matches or surpasses the rendering quality of existing methods across all evaluated datasets. Specifically, 3DCS outperforms 3DGS, GES and 2DGS in most metrics on the T&T and DB

datasets, while also achieving the second highest PSNR and lowest LPIPS on the Mip-NeRF360 dataset. 3DCS effectively balances memory usage and training time, sitting in between the ones of Mip-NeRF360 and 3DGS. Particularly, while it consumes more memory than Mip-NeRF360, it significantly reduces training time, requiring only 63 minutes compared to the 48 hours of Mip-NeRF360. Moreover, it delivers better visual quality, especially on the T&T dataset, where 3DCS demonstrates a notable performance advantage of over 1.73 PSNR compared to Mip-NeRF360. In comparison with 3DGS, 3DCS exhibits a slightly longer training time and lower rendering speed. Yet, 3DCS still operates within real-time rendering capabilities. Thanks to its greater adaptability, 3DCS efficiently utilizes only 70% of the memory needed by 3DGS, while achieving higher visual quality. Figure 7 strikes a qualitative comparison between 3DCS, 3DGS and 2DGS. Notably, our method achieves sharp and detailed rendering even in challenging regions, *e.g.* the background in the *Train* scene. In contrast, Gaussian primitives tend to oversmooth areas, resulting in images with pronounced artifacts, as observed in the *Flower*, *Bicycle*, and *Truck* scenes. The convex-based approach, however, produces results that closely align with the ground truth, showcasing higher fidelity and a superior ability to realistically represent 3D environments.

3DCS light outperforms 3DGS and GES on the T&T and DP dataset, while using less memory. Figure 8 contains a visual comparison between 3DGS and light 3DCS.

Indoor versus outdoor scenes. Table 2 presents a comparative analysis of indoor versus outdoor scenes from the Mip-NeRF360 dataset. Indoor scenes consist of structured, flat surfaces with hard edges, while outdoor scenes generally have more unstructured surfaces. This structural difference advantages convex shapes, which are better suited for capturing the geometric characteristics of indoor environments. In fact, for indoor scenes, it can be seen that 3DCS significantly outperforms 3DGS with an improvement of 0.9 PSNR, 0.007 SSIM, and 0.023 LPIPS, surpassing all other Gaussian-based methods. Moreover, 3DCS achieves
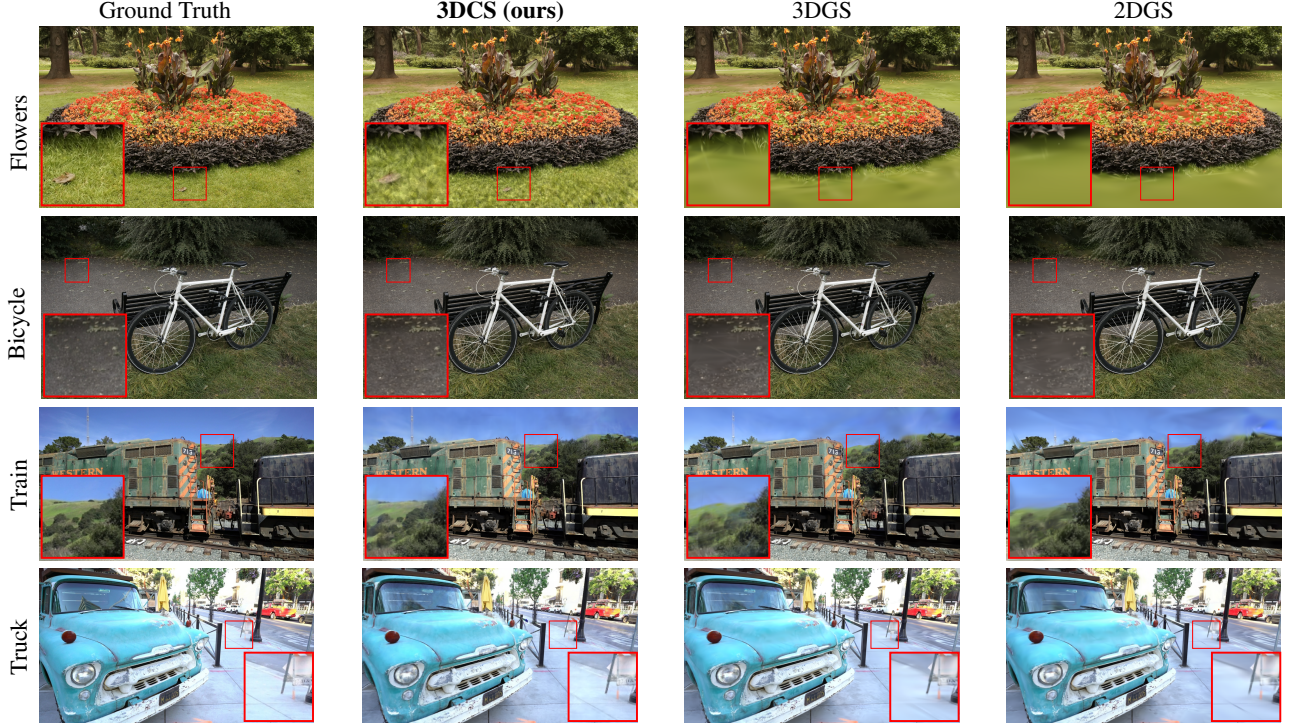
Figure 7. **Qualitative Comparison between 3DCS, 3DGS and 2DGS.** Our 3DCS captures finer details and provides a more accurate approximation of real-world scenes compared to Gaussian splatting methods, which often produce blurrier results.

	Outdoor Scene			Indoor scene		
	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑
MipNeRF360	0.283	24.47	0.691	0.180	31.72	0.917
3DGS	0.234	24.64	0.731	0.189	30.41	0.920
GES	0.243	24.46	0.724	0.189	30.85	0.922
2DGS	0.246	24.34	0.717	0.195	30.40	0.916
3DCS (ours)	0.238	24.07	0.700	0.166	31.33	0.927

Table 2. **Quantitative Results on Mip-NeRF 360 [3] Dataset.** We evaluate our method on both indoor and outdoor scenes, demonstrating substantial performance improvements over all 3DGS-based methods in indoor scenes and surpassing MipN-eRF360 in SSIM and LPIPS metrics.

Figure 8. **Visual Comparison Between our Light Model and 3DGS.** The light model (right) shows high visual quality compared to 3DGS (left), using less than 15% of the memory.

superior results in terms of SSIM and LPIPS metrics compared to Mip-NeRF360. Even in outdoor scenes containing a lot of human-made structures—such as the *Truck* and

Train scenes from T&T, 3DCS substantially outperforms 3DGS, demonstrating its ability to effectively handle structured geometries. However, in outdoor scenes dominated by nature and unstructured elements like trees and vegetation, the strengths of 3DCS become less pronounced. While 3DGS and 3DCS achieve comparable LPIPS results, 3DGS achieves better PSNR and SSIM. Yet, qualitatively, we can see in Fig. 7 that 3DCS appears significantly closer to the ground truth in terms of visual quality. Specifically, in the highlighted region of the *Flower* scene, our reconstruction better represents the real grass even though the PSNR of this area is 20.17 for 3DCS and 21.65 for 3DGS. This show-cases the popularly observed mismatch between PSNR and perceived visual quality. This is mainly due to the fact that PSNR is highly sensitive to pixel-level differences and, therefore, tends to favor blurrier images.

4.4. Ablation Study and Discussion

We analyze the impact of densification strategies, the number of points per convex shape, and the influence of reducing the number of shapes on rendering quality.

Densification strategy. We evaluate the effectiveness of splitting each convex shape into new convex shapes, as described in Sec. 3.3. Specifically, we analyze the impact of dividing a convex shape defined initially by $K = 6$ points into 2, 3, or 6 (default value) new convex shapes. For split-
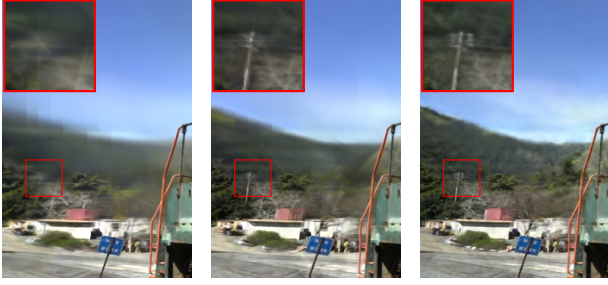
Figure 9. **Ablation of densification strategy.** From left to right, we split each convex into 2, 3, or 6 new convexes.

	$LPIPS^{\downarrow}$	$PSNR^{\uparrow}$	$SSIM^{\uparrow}$	Train$^{\downarrow}$
3DCS (K=3)	0.241	22.40	0.794	44m
3DCS (K=4)	0.159	23.73	0.848	52m
3DCS (K=5)	0.160	23.70	0.848	60m
3DCS (K=6)	**0.157**	**23.90**	0.850	71m
3DCS (K=7)	**0.157**	**23.90**	**0.851**	73m

Table 3. **Ablation Study of the Number of Points per Convex.** We study the impact of the number of points per convex on reconstruction quality and training time on the T&T dataset. With only 4 points, our 3DCS performs better than 3DGS.
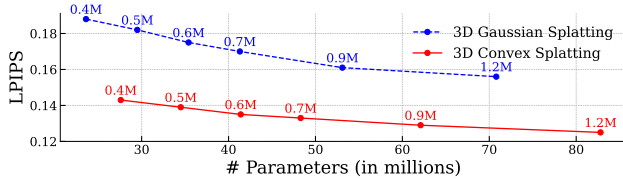

Figure 10. **# Parameters *vs*. $LPIPS^{\downarrow}$ (*Truck* scene).** The number of primitives is indicated for each point. 3DCS achieves a better regime than 3DGS for a comparable number of parameters.

ting into 2 or 3 shapes, the new convex shapes are centered on 2 or 3 randomly selected points from the original convex shape. As can be seen in Fig. 9, splitting a convex shape into more shapes results in higher visual quality, particularly in capturing finer details in the background.

Number of points per shape. Increasing K provides greater flexibility in representing convex shapes but comes at the cost of longer training times. Notably, the case of 3 points represents a special configuration, resulting in a non-volumetric triangle in 3D space, analogous to the 2D Gaussian Splatting approach [21] in terms of its dimensionality constraints. Table 3 shows that using $K \geq 4$ points per convex consistently outperforms 3DGS. However, increasing beyond 6 points has no significant performance gain.

Influence of less primitives on rendering quality. Figure 10 shows how LPIPS on the T&T dataset changes with the number of primitives and parameters. Notably, our

method 3DCS consistently outperforms 3DGS.

Physically meaningful 3D representations. Figure 11 visually compares the performance of 3DGS and 3DCS when the number of primitives N is reduced. With fewer shapes, 3DGS produces blurry images due to the limited flexibility of 3D Gaussians, which struggle to form visually meaningful representations of real objects. In contrast, 3DCS preserves image clarity by effectively decomposing objects into convex shapes. 3DCS represents the leaves on the stump as either a single convex shape or a collection of convex shapes, with each shape capturing a physically meaningful part of the real-world object. Ultimately, 3DCS offers a significant advantage by delivering more physically meaningful 3D representations. By leveraging the adaptability of convex shapes, we bridge the gap between visual accuracy and interpretability, enabling high-quality, geometrically meaningful 3D modeling.

Figure 11. **3DCS *vs*. 3DGS with fewer shapes.** Convex splatting (left) can decompose objects into meaningful convex shapes, enabling a realistic and compact 3D representation of the world.

5. Conclusion

We introduce 3D Convex Splatting (3DCS), a novel method for radiance field rendering that leverages 3D smooth convexes to achieve high-quality novel view synthesis. Particularly, our method overcomes the limitations of 3D Gaussian Splatting, delivering denser representations with fewer primitives and parameters. Furthermore, 3DCS demonstrates substantial improvements on the novel view synthesis task. By combining the adaptability of convex shapes with the efficiency of primitive-based radiance field rendering, 3DCS achieves high-quality, real-time, and flexible radiance field reconstruction. We envision this new primitive to set the ground for further research in the field.

References

[1] Sameer Agarwal, Yasutaka Furukawa, Noah Snavely, Ian Simon, Brian Curless, Steven M. Seitz, and Richard Szeliski. Building Rome in a day. *Commun. ACM*, 54(10):105–112, 2011. 1, 2

[2] 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 *IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, pages 5835–5844, Montréal, Can., 2021. 2

[3] Jonathan T. Barron, Ben Mildenhall, Dor Verbin, Pratul P. Srinivasan, and Peter Hedman. Mip-NeRF 360: Unbounded anti-aliased neural radiance fields. In *IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, pages 5460–5469, New Orleans, LA, USA, 2022. 2, 6, 7

[4] Jonathan T. Barron, Ben Mildenhall, Dor Verbin, Pratul P. Srinivasan, and Peter Hedman. Zip-NeRF: Anti-aliased grid-based neural radiance fields. In *IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, pages 19640–19648, Paris, Fr., 2023. 2, 6

[5] Eric R. Chan, Connor Z. Lin, Matthew A. Chan, Koki Nagano, Boxiao Pan, Shalini de Mello, Orazio Gallo, Leonidas Guibas, Jonathan Tremblay, Sameh Khamis, Tero Karras, and Gordon Wetzstein. Efficient geometry-aware 3D generative adversarial networks. In *IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, pages 16102–16112, New Orleans, LA, USA, 2022. 2

[6] Anpei Chen, Zexiang Xu, Andreas Geiger, Jingyi Yu, and Hao Su. TensoRF: Tensorial radiance fields. In *Eur. Conf. Comput. Vis. (ECCV)*, pages 333–350. Springer Nat. Switz., 2022. 2

[7] Zhiqin Chen, Andrea Tagliasacchi, and Hao Zhang. BSP-net: Generating compact meshes via binary space partitioning. In *IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, pages 42–51, Seattle, WA, USA, 2020. 3

[8] John H. Conway and Neil J. A. Sloane. *Sphere Packings, Lattices and Groups*. Springer New York, 1999. 2

[9] Boyang Deng, Kyle Genova, Soroosh Yazdani, Sofien Bouaziz, Geoffrey Hinton, and Andrea Tagliasacchi. CvxNet: Learnable convex decomposition. In *IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, pages 31–41, Seattle, WA, USA, 2020. 1, 2, 3, 4

[10] Robert A. Drebin, Loren Carpenter, and Pat Hanrahan. Volume rendering. In *ACM Int. Conf. Comput. Graph. Interact. Tech. (SIGGRAPH)*, pages 65–74, Atlanta, Georgia, USA, 1988. 2

[11] Yilun Du, Cameron Smith, Ayush Tewari, and Vincent Sitzmann. Learning to render novel views from wide-baseline stereo pairs. In *IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, pages 4970–4980, Vancouver, Can., 2023. 2

[12] Olivier D. Faugeras. What can be seen in three dimensions with an uncalibrated stereo rig? In *Eur. Conf. Comput. Vis. (ECCV)*, pages 563–578. Springer Berl. Heidelb., 1992. 1, 2

[13] Sara Fridovich-Keil, Alex Yu, Matthew Tancik, Qinhong Chen, Benjamin Recht, and Angjoo Kanazawa. Plenoxels: Radiance fields without neural networks. In *IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, pages 5491–5500, New Orleans, LA, USA, 2022. 2, 5

[14] Ronald L. Graham. An efficient algorith for determining the convex hull of a finite planar set. *Inf. Process. Lett.*, 1(4): 132–133, 1972. 3, 4

[15] Markus Gross and Hanspeter Pfister. *Point-Based Graphics*. Morgan Kauffmann Publ. Inc., 2007. 2

[16] Thomas C. Hales. The sphere packing problem. *J. Comput. Appl. Math.*, 44(1):41–76, 1992. 2

[17] Abdullah Hamdi, Bernard Ghanem, and Matthias Nießner. SPARF: Large-scale learning of 3D sparse radiance fields from few input images. In *IEEE/CVF Int. Conf. Comput. Vis. Work. (ICCV Work.)*, pages 2922–2932, Paris, Fr., 2023. 1

[18] 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 *IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, pages 19812–19822, Seattle, WA, USA, 2024. 3, 6

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

[20] Peter Hedman, Pratul P. Srinivasan, Ben Mildenhall, Jonathan T. Barron, and Paul Debevec. Baking neural radiance fields for real-time view synthesis. In *IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, pages 5855–5864, Montréal, Can., 2021. 2

[21] Binbin Huang, Zehao Yu, Anpei Chen, Andreas Geiger, and Shenghua Gao. 2D Gaussian splatting for geometrically accurate radiance fields. In *ACM SIGGRAPH Conf. Pap.*, pages 1–11, Denver, CO, USA, 2024. 3, 6, 8

[22] Yi-Hua Huang, Ming-Xian Lin, Yang-Tian Sun, Ziyi Yang, Xiaoyang Lyu, Yan-Pei Cao, and Xiaojuan Qi. Deformable radial kernel splatting. *arXiv*, abs/2412.11752, 2024. 3

[23] Ajay Jain, Matthew Tancik, and Pieter Abbeel. Putting NeRF on a diet: Semantically consistent few-shot view synthesis. In *IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, pages 5865–5874, Montréal, Can., 2021. 2

[24] James T. Kajiya and Brian P. Von Herzen. Ray tracing volume densities. *ACM SIGGRAPH Comput. Graph.*, 18(3): 165–174, 1984. 2

[25] Hiroharu Kato, Yoshitaka Ushiku, and Tatsuya Harada. Neural 3D mesh renderer. In *IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, pages 3907–3916, Salt Lake City, UT, USA, 2018. 2

[26] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkuehler, and George Drettakis. 3D Gaussian splatting for real-time radiance field rendering. *ACM Trans. Graph.*, 42(4):1–14, 2023. 1, 2, 3, 4, 5, 6

[27] Mijeong Kim, Seonguk Seo, and Bohyung Han. InfoNeRF: Ray entropy minimization for few-shot neural volume rendering. In *IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, pages 12902–12911, New Orleans, LA, USA, 2022. 2

[28] Arno Knapitsch, Jaesik Park, Qian-Yi Zhou, and Vladlen Koltun. Tanks and temples: benchmarking large-scale scene reconstruction. *ACM Trans. Graph.*, 36(4):1–13, 2017. 6

[29] Joo Chan Lee, Daniel Rho, Xiangyu Sun, Jong Hwan Ko, and Eunbyung Park. Compact 3D Gaussian representation for radiance field. In *IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, pages 21719–21728, Seattle, WA, USA, 2024. 5, 6

[30] Marc Levoy. Efficient ray tracing of volume data. *ACM Trans. Graph.*, 9(3):245–261, 1990. 2

[31] Hsueh-Ti Derek Liu, Michael Tao, and Alec Jacobson. Paparazzi: surface editing by way of multi-view image processing. *ACM Trans. Graph.*, 37(6):1–11, 2018. 2

[32] Shichen Liu, Weikai Chen, Tianye Li, and Hao Li. Soft rasterizer: A differentiable renderer for image-based 3D reasoning. In *IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, pages 7707–7716, Seoul, South Korea, 2019.

[33] Matthew M. Loper and Michael J. Black. OpenDR: An approximate differentiable renderer. In *Eur. Conf. Comput. Vis. (ECCV)*, pages 154–169. Springer Int. Publ., 2014. 2

[34] Jonathon Luiten, Georgios Kopanas, Bastian Leibe, and Deva Ramanan. Dynamic 3D Gaussians: Tracking by persistent dynamic view synthesis. In *Int. Conf. 3D Vis. (3DV)*, pages 800–809, Davos, Switzerland, 2024. 3

[35] Alexander Mai, Peter Hedman, George Kopanas, Dor Verbin, David Futschik, Qiangeng Xu, Falko Kuester, Jonathan T. Barron, and Yinda Zhang. EVER: Exact volumetric ellipsoid rendering for real-time view synthesis. *arXiv*, abs/2410.01804, 2024. 3

[36] Jinjie Mai, Wenxuan Zhu, Sara Rojas, Jesus Zarzar, Abdullah Hamdi, Guocheng Qian, Bing Li, Silvio Giancola, and Bernard Ghanem. TrackNeRF: Bundle adjusting NeRF from sparse and noisy views via feature tracks. In *Eur. Conf. Comput. Vis. (ECCV)*, Milan, Italy, 2024. 1

[37] Nelson Max. Optical models for direct volume rendering. *IEEE Trans. Vis. Comput. Graph.*, 1(2):99–108, 1995. 2

[38] 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. In *Eur. Conf. Comput. Vis. (ECCV)*, pages 405–421. Springer Int. Publ., 2020. 1, 2

[39] Thomas Müller, Alex Evans, Christoph Schied, and Alexander Keller. Instant neural graphics primitives with a multiresolution hash encoding. *ACM Trans. Graph.*, 41(4):1–15, 2022. 2, 6

[40] Richard A. Newcombe, Andrew Fitzgibbon, Shahram Izadi, Otmar Hilliges, David Molyneaux, David Kim, Andrew J. Davison, Pushmeet Kohi, Jamie Shotton, and Steve Hodges. KinectFusion: Real-time dense surface mapping and tracking. In *IEEE Int. Symp. Mix. Augment. Real.*, pages 127–136, Basel, Switzerland, 2011. 1

[41] Simon Niedermayr, Josef Stumpfegger, and Rüdiger Westermann. Compressed 3D Gaussian splatting for accelerated novel view synthesis. In *IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, pages 10349–10358, Seattle, WA, USA, 2024. 6

[42] Despoina Paschalidou, Ali Osman Ulusoy, and Andreas Geiger. Superquadrics revisited: Learning 3D shape parsing beyond cuboids. In *IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, pages 10336–10345, Long Beach, CA, USA, 2019. 3

[43] Felix Petersen, Amit H. Bermano, Oliver Deussen, and Daniel Cohen-Or. Pix2Vex: Image-to-geometry reconstruction using a smooth differentiable renderer. *arXiv*, abs/1903.11149, 2019. 2

[44] Franco P. Preparata and Sie June Hong. Convex hulls of finite sets of points in two and three dimensions. *Commun. ACM*, 20(2):87–93, 1977. 3

[45] Christian Reiser, Songyou Peng, Yiyi Liao, and Andreas Geiger. KiloNeRF: Speeding up neural radiance fields with thousands of tiny MLPs. In *IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, pages 14315–14325, Montréal, Can., 2021. 2

[46] Christian Reiser, Rick Szeliski, Dor Verbin, Pratul Srinivasan, Ben Mildenhall, Andreas Geiger, Jon Barron, and Peter Hedman. MERF: Memory-efficient radiance fields for real-time view synthesis in unbounded scenes. *ACM Trans. Graph.*, 42(4):1–12, 2023. 2

[47] Daxuan Ren, Haiyi Mei, Hezi Shi, Jianmin Zheng, Jianfei Cai, and Lei Yang. Differentiable convex polyhedra optimization from multi-view images. In *Eur. Conf. Comput. Vis. (ECCV)*, pages 251–269. Springer Nat. Switz., 2024. 3

[48] Cheng Sun, Min Sun, and Hwann-Tzong Chen. Direct voxel grid optimization: Super-fast convergence for radiance fields reconstruction. In *IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, pages 5449–5459, New Orleans, LA, USA, 2022. 2

[49] Shubham Tulsiani, Hao Su, Leonidas J. Guibas, Alexei A. Efros, and Jitendra Malik. Learning shape abstractions by assembling volumetric primitives. *arXiv*, abs/1612.00404, 2016. 3

[50] Dor Verbin, Peter Hedman, Ben Mildenhall, Todd Zickler, Jonathan T. Barron, and Pratul P. Srinivasan. Ref-NeRF: Structured view-dependent appearance for neural radiance fields. In *IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, pages 5481–5490, New Orleans, LA, USA, 2022. 2

[51] Xinyue Wei, Minghua Liu, Zhan Ling, and Hao Su. Approximate convex decomposition for 3D meshes with collision-aware concavity and tree search. *arXiv*, abs/2205.02961, 2022. 3

[52] Lior Yariv, Peter Hedman, Christian Reiser, Dor Verbin, Pratul P. Srinivasan, Richard Szeliski, Jonathan T. Barron, and Ben Mildenhall. BakedSDF: Meshing neural SDFs for real-time view synthesis. In *ACM SIGGRAPH Conf. Proc.*, pages 1–9, Los Angeles, CA, USA, 2023. 2

[53] Alex Yu, Vickie Ye, Matthew Tancik, and Angjoo Kanazawa. pixelNeRF: Neural radiance fields from one or few images. In *IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, pages 4576–4585, Nashville, TN, USA, 2021. 2

[54] Zehao Yu, Anpei Chen, Binbin Huang, Torsten Sattler, and Andreas Geiger. Mip-splatting: Alias-free 3D Gaussian splatting. In *IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, pages 19447–19456, Seattle, WA, USA, 2024. 3

[55] Kai Zhang, Gernot Riegler, Noah Snavely, and Vladlen Koltun. NeRF++: Analyzing and improving neural radiance fields. *arXiv*, abs/2010.07492, 2020. 2

3D Convex Splatting: Radiance Field Rendering with 3D Smooth Convexes

Supplementary Material

6. Hyperparameters & Initialization

We initialize each convex shape using the COLMAP points as centers and uniformly distribute *new* points around them with the Fibonacci sphere algorithm to ensure an even distribution. The initial sphere radius is set to 1.2 times the mean distance to the three nearest neighbors in the point cloud. This adaptive initialization ensures that dense 3D regions contain many small convex shapes, while sparser regions are represented by larger convexes. The initial values for the smoothness parameter δ and sharpness parameter σ are set to 0.1 and $9.5e^{-4}$, respectively. These values are chosen to initially produce more diffuse shapes, as this configuration was empirically found to result in better performance during optimization. The initial opacity is set to 0.1. A detailed list of the hyperparameters can be found in Tab. 4.

7. Methodology Details

2D equations. We define the 2D convex indicator function for our convex hull by adapting the smooth convex representation from 3D to 2D, utilizing the equations introduced in Sec. 3.1. Specifically, we define $\phi(\mathbf{q})$ and $I(\mathbf{q})$ as in Eqs. (2) and (3), but substitute the 3D point $\mathbf{p}$ with the 2D point $\mathbf{q}$ and replace the planes delimiting the 3D convex hull with the lines that delimit the resulting 2D convex hull.

$$\phi(\mathbf{q}) = \log \left(\sum_{t=1}^{T} \exp \left(d \, \delta \, L_j(\mathbf{q}) \right) \right), \qquad (7)$$

where T is the total number of lines delimiting the 2D convex shape.

The indicator function $I(\mathbf{q})$ of the smooth convex is then defined by:

$$I(\mathbf{q}) = \text{Sigmoid} \left(-d \, \sigma \, \phi(\mathbf{q}) \right). \qquad (8)$$

8. Ablation Study

Perspective-Aware Scaling in 2D Projection. To incorporate perspective effects in the 2D projection, we scale δ and σ by the distance d, ensuring that the appearance of the convex shape remains consistent regardless of its distance from the camera. Table 5 provides an ablation study demonstrating the necessity of scaling δ and σ as well as analyzing the impact of the scaling magnitude.

9. More Results

Experiments on Synthetic Data. To highlight the effectiveness of our smooth convexes for rendering different shapes, we designed four 2D synthetic examples, where the optimization is performed in 2D, with a single image as ground truth. Figure 12 illustrates the optimization process of our smooth convexes on those four distinct shapes during training. Our convex shapes are highly versatile and capable of approximating a wide range of different shapes.

Real-world Novel View Synthesis. Figure 13 shows additional qualitative results, highlighting the capabilities of 3D Convex Splatting compared to 3D Gaussians and 2D Gaussians. The inherent softness of Gaussian primitives often leads to blurrier images and noticeable artifact-prone regions. While PSNR favors such blurrier images due to imprecise image alignment, they lack high-quality detail. Compared to Gaussian, 3D Convex Splatting does not produce any blurry areas and often results in a rendering much closer to the ground truth. For instance, in the *Bicycle* scene, Gaussian methods produce blurry artifacts on the street and in the grass, whereas 3D Convex Splatting achieves a result that closely matches the ground truth. Tables 6 to 11 show the complete quantitative results for each scene. 3D Convex Splatting outperforms 3DGS, 2DGS, and GES across all metrics on indoor scenes, the Deep Blending dataset, and the Tanks & Temples dataset.

Additional Visualizations. Figure 14 shows the depth maps obtained using 3D convexes *vs*. Gaussians, illustrating our better geometric quality, producing fewer artifacts and smoother depths. Furthermore, Fig. 15 shows the representation of a wall. It confirms that convexes generate denser and more physically accurate wall representations, whereas Gaussians produce more diffused shapes.

Table 4. Hyperparameters for Convex Splatting

Parameter	Indoor		Outdoor	
	Normal	**Light**	**Normal**	**Light**
Number of training iterations		30, 000		
position_lr_delay_mult		0.01		
position_lr_max_steps		30, 000		
feature_lr		0.0025		
opacity_lr		0.01		
lambda_dssim		0.2		
densification_interval		200		
densify_from_iter		500		
opacity_reset		0.2		
opacity_reset_interval		3000		
remove_size_threshold		0.3		
min_opacity		0.03		
lr_mask		0.01		
lr_delta		0.005		
nb_points		6		
convex_size		1.2		
set_opacity		0.1		
set_delta		0.1		
opacity_cloning		0.5		
delta_scaling_cloning		1		
shifting_cloning		1		
set_sigma	$9\,e^{-4}$	$9.5\,e^{-4}$	$1\,e^{-3}$	$9.5\,e^{-4}$
densify_grad_threshold	$6\,e^{-6}$	$2.5\,e^{-5}$	$1\,e^{-6}$	$2.5\,e^{-5}$
lr_sigma	$4.5\,e^{-3}$	$4.5\,e^{-3}$	$4\,e^{-3}$	$4.5\,e^{-3}$
reset_opacity_until	5000	5000	18000	5000
scaling_cloning	0.7	0.63	0.6	0.63
sigma_scaling_cloning	0.85	0.88	0.88	0.88
mask_threshold	$2\,e^{-2}$	$1\,e^{-2}$	$1\,e^{-2}$	$1\,e^{-2}$
lr_convex_points_init	$4\,e^{-4}$	$5\,e^{-4}$	$5\,e^{-4}$	$5\,e^{-4}$
lr_convex_points_final	$4\,e^{-6}$	$5\,e^{-6}$	$5\,e^{-6}$	$5\,e^{-6}$
densify_until_iter	9500	9000	9000	9000
storage	32-bit precision	16-bit precision	32-bit precision	16-bit precision

Magnitude	Truck	Train	DrJohnson	Playroom
1	19.47	19.14	29.17	28.82
$\sqrt{d}$	25.49	21.41	29.49	29.98
d	25.65	22.23	29.54	30.08
d^2	7.08	8.91	8.42	8.99

Table 5. **Perspective-Aware Scaling in 2D Projection.** We evaluate the PSNR under varying scaling magnitudes.

	Truck	Train	DrJohnson	Playroom
3DGS	0.148	0.218	0.244	0.241
2DGS	0.173	0.251	0.257	0.257
GES	0.162	0.232	0.249	0.252
3DCS	0.125	0.187	0.238	0.237

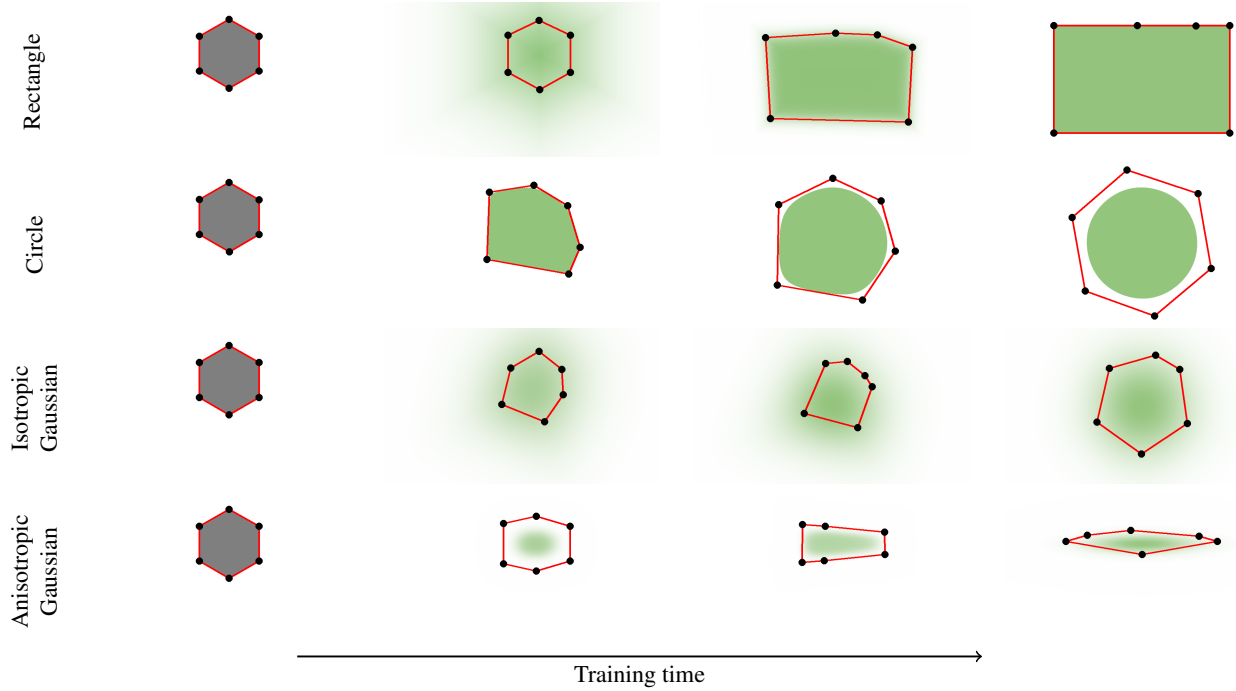Table 6. LPIPS score for T&T and DB datasets.

Figure 12. Smooth convexes can represent a wide variety of shapes, whether hard or soft, dense or diffuse. They effectively approximate diverse geometries, including both polyhedra and Gaussians, while requiring fewer primitives for accurate representation. The red lines describe the convex hull, whereas the black dots represent the point set.

	Truck	Train	DrJohnson	Playroom
3DGS	25.18	21.09	28.76	30.04
2DGS	25.12	21.14	28.95	30.05
GES	25.07	21.75	29.24	30.06
3DCS	25.65	22.23	29.54	30.08

Table 7. PSNR score for T&T and DB datasets.

	Truck	Train	DrJohnson	Playroom
3DGS	0.879	0.802	0.899	0.906
2DGS	0.874	0.789	0.900	0.906
GES	0.872	0.800	0.899	0.902
3DCS	0.882	0.820	0.902	0.902

Table 8. SSIM score for T&T and DB datasets.

	Bicycle	Flowers	Garden	Stump	Treehill	Room	Counter	Kitchen	Bonsai
3DGS	0.205	0.336	0.103	0.210	0.317	0.220	0.204	0.129	0.205
2DGS	0.218	0.346	0.115	0.222	0.329	0.223	0.208	0.133	0.214
GES	0.272	0.342	0.110	0.218	0.331	0.220	0.202	0.127	0.206
3DCS	0.216	0.322	0.113	0.227	0.317	0.193	0.182	0.117	0.182

Table 9. LPIPS score for the MipNerf360 dataset.

	Bicycle	Flowers	Garden	Stump	Treehill	Room	Counter	Kitchen	Bonsai
3DGS	25.24	21.52	27.41	26.55	22.49	30.63	28.70	30.31	31.98
2DGS	24.87	21.15	26.95	26.47	22.27	31.06	28.55	30.50	31.52
GES	24.76	21.33	26.89	26.06	22.31	31.03	28.88	31.21	31.94
3DCS	24.72	20.52	27.09	26.12	21.77	31.70	29.02	31.96	32.64

Table 10. PSNR score for the MipNerf360 dataset.

	Bicycle	Flowers	Garden	Stump	Treehill	Room	Counter	Kitchen	Bonsai
3DGS	0.771	0.605	0.868	0.775	0.638	0.914	0.905	0.922	0.938
2DGS	0.752	0.588	0.852	0.765	0.627	0.912	0.900	0.919	0.933
GES	0.727	0.600	0.846	0.768	0.631	0.910	0.899	0.920	0.939
3DCS	0.737	0.575	0.850	0.746	0.595	0.925	0.909	0.930	0.945

Table 11. SSIM score for the MipNerf360 dataset.

Figure 13. **Qualitative Comparison between 3DCS, 3DGS and 2DGS.** 3D Convex Splatting achieves high-quality novel view synthesis and fast rendering by representing scenes with 3D smooth convexes. In contrast, the softness of Gaussian primitives often results in blurring and loss of detail, while 3D Convex Splatting effectively captures sharp edges and fine details.
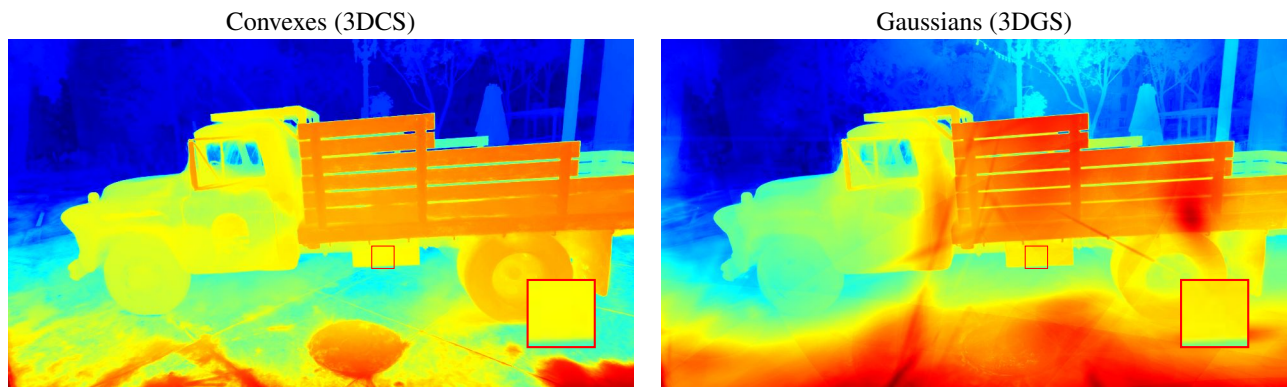
Figure 14. **Depthmap Comparison.** 3DCS generates fewer artifacts and a cleaner depth map than 3DGS.
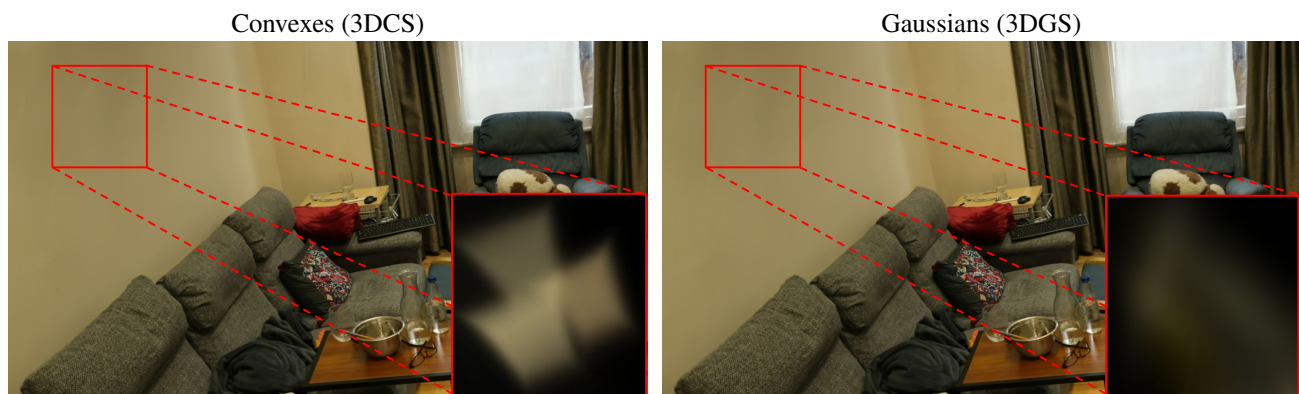

Figure 15. **Wall Approximation.** Example showing that 3DCS approximates a wall more uniformly and accurately than 3DGS.