UniMGS: Unifying Mesh and 3D Gaussian Splatting with Single-Pass Rasterization and Proxy-Based Deformation

Zeyu Xiao[1*], Mingyang Sun[1*], Yimin Cong[1], Lintao Wang[1], Dongliang Kou[1], Zhenyi Wu[1], Dingkang Yang[1], Peng Zhai[1], Zeyu Wang[3, 4†], Lihua Zhang[1, 2†]

[1]College of Intelligent Robotics and Advanced Manufacturing, Fudan University, Shanghai, China
[2]Fysics Intelligence Technologies Co., Ltd., Shanghai, China
[3]The Hong Kong University of Science and Technology (Guangzhou), Guangzhou, China
[4]The Hong Kong University of Science and Technology, Hong Kong, China

Abstract

Joint rendering and deformation of mesh and 3D Gaussian Splatting (3DGS) have significant value as both representations offer complementary advantages for graphics applications. However, due to differences in representation and rendering pipelines, existing studies render meshes and 3DGS separately, making it difficult to accurately handle occlusions and transparency. Moreover, the deformed 3DGS still suffers from visual artifacts due to the sensitivity to the topology quality of the proxy mesh. These issues pose serious obstacles to the joint use of 3DGS and meshes, making it difficult to adapt 3DGS to conventional mesh-oriented graphics pipelines. We propose UniMGS, the first unified framework for rasterizing mesh and 3DGS in a single-pass anti-aliased manner, with a novel binding strategy for 3DGS deformation based on proxy mesh. Our key insight is to blend the colors of both triangle and Gaussian fragments by anti-aliased α-blending in a single pass, achieving visually coherent results with precise handling of occlusion and transparency. To improve the visual appearance of the deformed 3DGS, our Gaussian-centric binding strategy employs a proxy mesh and spatially associates Gaussians with the mesh faces, significantly reducing rendering artifacts. With these two components, UniMGS enables the visualization and manipulation of 3D objects represented by mesh or 3DGS within a unified framework, opening up new possibilities in embodied AI, virtual reality, and gaming. We will release our source code to facilitate future research.

Introduction

Recent advances in radiance representations, especially 3D Gaussian Splatting (3DGS) (Kerbl et al. 2023), have enabled the efficient modeling of photorealistic 3D scenes. Adding meshes into a radiance scene for interaction and visualization has shown great value in applications like embodied AI (Xie et al. 2025) and gaming (Xia et al. 2024). Consequently, there is an urgent need in the 3D vision community to improve the compatibility between 3DGS and mesh for both rendering and manipulation.

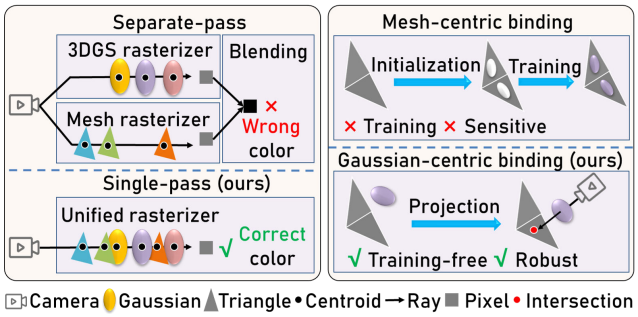


Figure 1: Differences between UniMGS and prior work in rasterization and deformation.

To account for the differences in the rendering between the shape representations, most methods first render meshes and 3DGS separately and then blend the results via visibility and depth sorting, enabling a straightforward separate-pass implementation (Xie et al. 2025; Wang et al. 2024; Xiao et al. 2024). Since the separate-pass scheme must render each shape individually to calculate visibility, it is inherently unable to handle nested spatial relationships between objects. Instead, a single-pass method enables the simultaneous processing of both meshes and a radiance representation within a unified rendering pipeline (Qiao et al. 2023), which remains insufficiently explored. Although ray tracing has recently allowed for rendering 3DGS and meshes (Byrski et al. 2025; Wu et al. 2025), its run-time performance remains inferior to rasterization due to high computational cost.

In addition, since shapes represented by 3DGS consist of discrete Gaussian kernels with center points, deformation can be achieved by directly manipulating them (Wu et al. 2024; Xie et al. 2024). An alternative approach deforms 3DGS indirectly by attaching it to a proxy and manipulating the proxy, e.g., sparse control points (Huang et al. 2024; Zhong et al. 2024), deformation graphs (Tong et al. 2025a), cages (Tong et al. 2025b; Huang and Yu 2024), meshes (Gao et al. 2024a; Waczyńska et al. 2024; Gao et al. 2024b). As mentioned in GaussianMesh (Gao et al. 2024a), paradigms other than those based on mesh proxies lack explicit topological information, leading to misalignment ar-

*These authors contributed equally.
†These authors are equal corresponding authors.

tifacts when handling large deformations. However, existing approaches that rely on proxy meshes typically adopt a mesh-centric binding strategy, where Gaussians are first initialized on each face and then refined through training with geometric constraints from the proxy mesh. As they follow an "initialization-then-training" workflow, even an optimized 3DGS must be retrained to bind with a proxy mesh, which forms a causal dependency. This design not only reduces usability but also is sensitive to the mesh's topology, leading to artifacts when the mesh contains defects.

Motivated by these observations, we propose **UniMGS**, a **Uni**fied framework for **M**esh and 3D**GS** integration in single-pass rasterization and proxy-based deformation. Our core idea consists of two components: 1) adapting the 3DGS rasterizer for anti-aliased mesh rendering; and 2) adapting mesh-oriented manipulation to 3DGS. Since both 3DGS and meshes are rasterizable, directly assigning an opacity attribute to textured meshes allows for unified rendering through α-blending. However, the implementation of anti-aliasing differs between the two approaches: for 3DGS, the Elliptical Weighted Average (EWA) filter (Zwicker et al. 2001; Kerbl et al. 2023) is applied before α-blending. In contrast, for triangle meshes, anti-aliasing is typically performed during α-blending and involves a coverage value that depends on the sub-pixel transmittance (de Vries 2020). Therefore, the α-blending in 3DGS must be adapted to account for anti-aliasing of meshes. Our method emerges naturally from this line of reasoning. We treat the depth-adjacent triangles as a single entity, within which color blending is still performed at the pixel level, while transmittance is independently computed at the sub-pixel level and weighted by coverage to represent pixel-level transmittance. Outside the entity, each triangle is still considered an individual fragment, since anti-aliasing only applies within this triangle entity. Therefore, transmittance must be updated among each triangle fragment at the pixel level to contribute to subsequent calculations, such as blending with Gaussians or the background. This novel design successfully integrates anti-aliased mesh rendering into the 3DGS rasterizer without any degradation, thereby enabling Gaussians and meshes to be rasterized together in a single-pass manner.

To adapt mesh-oriented manipulation to 3DGS, we propose a novel Gaussian deformation representation that enables topology-aware manipulation of 3DGS. Given the 3DGS of an object and its mesh, which is easily acquired via various reconstruction methods (Yu, Sattler, and Geiger 2024; Wang et al. 2023), our goal is to link each Gaussian to the mesh faces to allow propagation of the deformation. To enhance usability and deformation robustness, we advocate a Gaussian-centric perspective to eliminate the causal dependency involved in the mesh-centric ones. The spatial relationship between Gaussians and mesh faces can be directly utilized for binding. Thanks to cameras in the training dataset, we propose using ray casting as an accurate and efficient solution. Specifically, a ray is cast from a camera through the center of a Gaussian; if it intersects a face of the proxy mesh, we record that face. After traversing all cameras, multiple faces may be obtained. We select the nearest face as the binding target. During manipulation, deformation is transferred from the mesh to the Gaussian. We further extend this method to the Bounding Box (BBX) of Gaussians to enhance deformation robustness, as detailed later. Compared to mesh-centric designs, our Gaussian-centric strategy naturally leverages spatial relationships to associate Gaussians with faces without retraining, thereby achieving higher robustness to mesh imperfections during deformation.

We extensively evaluate *UniMGS* across diverse experiments. It achieves the first unified rasterization of meshes and 3DGS with faithful transparency, color, and correct occlusion handling. Compared to existing mesh-centric paradigms, our Gaussian-centric binding strategy delivers superior visual quality of deformation, even with a flawed proxy mesh, significantly outperforming prior methods. *UniMGS* bridges the gap between 3DGS and mesh while maintaining high flexibility in use. Both the rasterization and deformation modules independently improve the compatibility between 3DGS and mesh, and their integration further empowers simulations under the hybrid representations.

Related Work

Rendering Methods for Hybrid Representations

There are two primary paradigms for rendering 3D primitives onto an image plane: ray-based (e.g., ray marching, ray tracing) and primitive-based (rasterization) (Hughes et al. 2013), both of which are commonly used for meshes. The Neural Radiance Field (NeRF) (Mildenhall et al. 2020) adopts ray marching to enable volumetric rendering, while 3DGS (Kerbl et al. 2023) utilizes rasterization for fast rendering. Despite sharing similar rendering principles, bridging different shape representations within a unified rendering framework remains challenging. Most studies render the mesh separately first, then use its depth map to account for occlusion when rendering the radiance representation. In mesh-NeRF hybrid rendering, it is commonly assumed that the mesh is opaque, allowing ray marching to terminate early once the mesh surface is hit (Ye et al. 2024; Guo et al. 2023; Xia et al. 2024). In mesh-3DGS hybrid rendering, existing studies (Xiao et al. 2024; Wang et al. 2024; Xie et al. 2025) first rasterize meshes and then combine the rasterized result with the rendered Gaussians by α-blending. Instead, DMERF (Qiao et al. 2023) achieves mesh-NeRF coupling in a single-pass manner by allowing rays to alternate between ray tracing and ray marching. It is evident that single-pass approaches offer substantial advantages over separate-pass methods in accurately calculating transparency and occlusion. Recent studies (Moenne-Loccoz et al. 2024; Wu et al. 2025; Byrski et al. 2025) have proposed using ray tracing to render 3DGS, enabling seamless integration with meshes. While they improve visual quality, it comes at the cost of nearly halving the rendering speed.

Representation for 3DGS Deformation

3DGS (Kerbl et al. 2023) is a discrete shape representation with each Gaussian continuously parameterized by the kernel function. Though directly manipulating 3DGS is feasible (Xie et al. 2024; Wu et al. 2024), deformations often suffer from artifacts due to the lack of topological in-

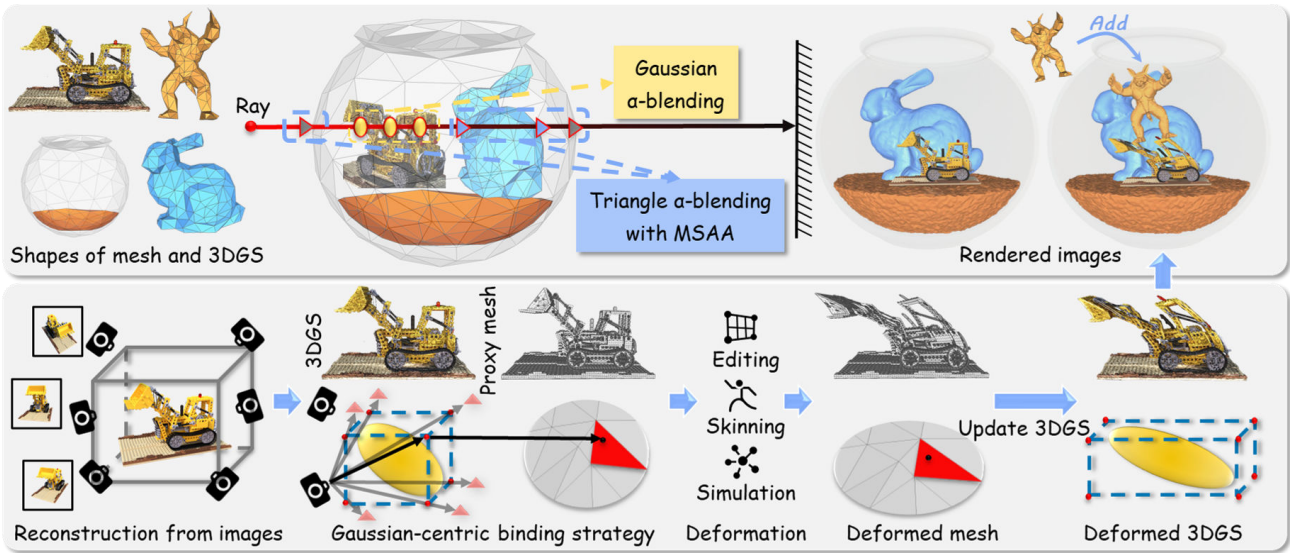


Figure 2: Overview of UniMGS. Given objects composed of 3DGS and mesh, the unified rendering pipeline bridges Gaussians and triangles by α-blending in a single-pass manner, thus accurately computing color and handling occlusion. To mitigate aliasing artifacts, we group depth-adjacent triangle fragments as a single entity, in which α-blending with MSAA is performed. The framework further allows mesh-based deformation to be seamlessly extended to 3DGS with a proxy mesh. We associate a Gaussian with triangle faces by projecting the vertices of its BBX onto the mesh. During deformation, the motion of triangle faces is first transferred to the BBX and then propagated to the Gaussian.

formation. Therefore, recent studies have proposed proxy-based representations that enable indirect deformation of 3DGS, such as sparse control points (Huang et al. 2024; Zhong et al. 2024; Li, Chen, and Liu 2024), deformation graph (Tong et al. 2025a), cage (Tong et al. 2025b; Huang and Yu 2024), and mesh (Gao et al. 2024a; Waczyńska et al. 2024). Among them, the proxy mesh performs better due to its complete topology and compatibility with most mesh-oriented manipulation methods, as confirmed by recent advanced research (Gao et al. 2024a,b). Given a proxy mesh, SuGaR (Guédon and Lepetit 2024b) and GaMeS (Waczyńska et al. 2024) attach multiple Gaussians to each triangle face, then these Gaussians move with the triangle vertices weighted by barycentric coordinates. Frosting (Guédon and Lepetit 2024a) forms a frosting layer around the mesh, where prismatic cells are created to enclose the Gaussians and control the Gaussians' motion during deformation. Mani-GS (Gao et al. 2024b) places Gaussians in the local coordinate system of mesh faces and drives the Gaussians through mesh editing. GaussianMesh (Gao et al. 2024a) assigns a single Gaussian to each triangle face, while they split together during training, and performs ACAP deformation (Gao et al. 2021) on the mesh to edit Gaussians. The above methods first initialize Gaussians on mesh faces through a mesh-centric binding scheme, then perform training, which makes the deformed Gaussians prone to visual artifacts due to their sensitivity to the topology quality of the proxy mesh.

Departing from previous studies, we propose a single-pass rasterization pipeline for 3DGS and mesh based on α-blending, which facilitates transparency rendering and correct spatial occlusion relationship without dropping rendering rates. In terms of proxy-based 3DGS deformation, we suggest a different insight by proposing a Gaussian-centric strategy that directly links trained 3DGS to the proxy mesh, resulting in minimal artifacts and robustness to mesh quality. Collectively, these contributions promote significant progress in the visualization and manipulation of mesh and 3DGS within dynamic scenes.

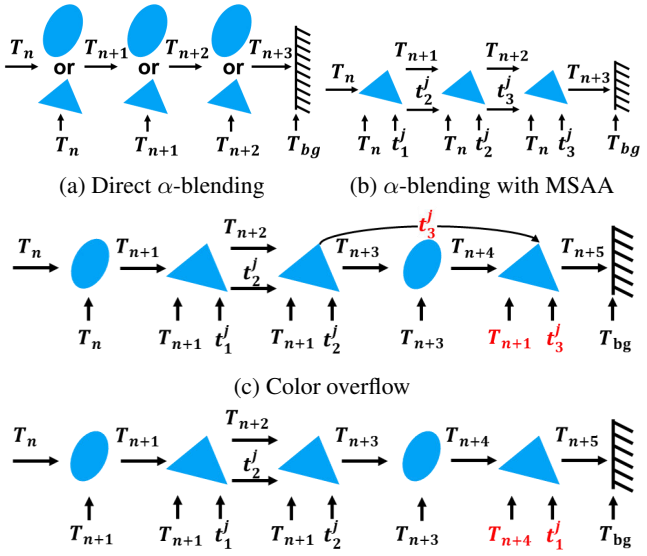
Methodology

Framework Overview

Figure 2 outlines our unified framework, which consists of two key modules: rasterization and deformation. We begin by reconstructing the 3DGS from multi-view images, while the corresponding proxy mesh can be obtained through various reconstruction algorithms (Wang et al. 2023; Yu, Sattler, and Geiger 2024; Kazhdan, Bolitho, and Hoppe 2006). Then, we apply the Gaussian-centric binding strategy to associate each Gaussian with relevant mesh faces. The deformation is initially performed on the proxy mesh and subsequently propagated to the 3DGS. For rasterization of different objects represented by 3DGS or meshes, we integrate anti-aliased triangle α-blending to the 3DGS rendering pipeline. This single-pass pipeline ensures correct handling of occlusion and color blending. Next, we introduce the rasterization and deformation modules in detail.

Rasterizing 3DGS and Mesh in a Single Pass

To better clarify our contribution on rasterization, we provide additional diagrams and rendering results alongside the equations. Figure 3 depicts how our unified rendering



(a) Direct α-blending (b) α-blending with MSAA

(c) Color overflow

(d) Our full method

Figure 3: Illustration of single-pass rasterization with α-blending. $\rightarrow$ means the update of transmittance while $\uparrow$ represents the transmittance used in α-blending. (a): Directly blend triangles and Gaussians without anti-aliasing of triangles. (b): Consider all triangles overlapping the same pixel as a whole and perform MSAA in α-blending. (c): Apply (b) directly to (a) causes color overflow. (d): Modify (c) by treating only depth-adjacent triangles as an entity, where the difference is highlighted in red.

method is constructed, while Figure 4 visualizes each part to justify the rationale behind each step of our method.

The 3DGS rasterizer employs α-blending that the color C_{gs} of a pixel is computed by blending N depth-ordered fragments overlapping the pixel along a ray (Kerbl et al. 2023):

$$C_{\text{gs}} = \sum_{i=1}^{N} T_i \alpha_i c_i, \tag{1}$$

with

$$T_i = \prod_{j=1}^{i-1} (1 - \alpha_j) \text{ and } T_1 = T_{\text{in}}. \tag{2}$$

In Equation (1), c_i and α_i are the RGB value and the opacity of the i-th fragment, T_i is called transmittance and represents the visibility of the i-th fragment (Maule et al. 2012), and T_{in} denotes the transmittance before the ray passes through these fragments, typically initialized to 1.0 if no fragments have been traversed. For unified rasterization, a straightforward approach is to incorporate triangle fragments into the depth-sorting process, enabling seamless integration into the 3DGS rasterizer (see Figure 3a and 4a). However, while Gaussians alleviate aliasing through EWA filtering, it is also essential to equip triangle fragments with an anti-aliasing strategy. Multi-Sample Anti-Aliasing (MSAA) (de Vries 2020) is a popular option (Epic Games, Inc. 2025b) that scales an opaque triangle fragment's color contribution by its coverage on sub-pixels:

$$C_{\text{tris}} = O_i c_i, \tag{3}$$

with

$$O_i = \frac{1}{M} \sum_{j=1}^{M} \hat{o}_i^j \text{ and } \hat{o}_i^j = \begin{cases} o_i^j, & o_{i-1} = 0 \\ 0, & o_{i-1} = 1 \end{cases}, \tag{4}$$

where $o_i^j \in \{0, 1\}$ represents whether the i-th fragment geometrically covers the j-th sub-pixel, $\hat{o}_i^j$ is the coverage after depth-testing, O_i is the total coverage at pixel level, and M is the number of sub-pixels (we set M=4). As mentioned in (Maule et al. 2012), a common approach combines MSAA with α-blending by:

$$C_{\text{tris}} = \sum_{i=1}^{N} T_i O_i \alpha_i c_i, \tag{5}$$

with

$$O_i = \frac{1}{M} \sum_{j=1}^{M} o_i^j \text{ and } T_i = \prod_{k=1}^{i-1} (1 - O_k \alpha_k). \tag{6}$$

In Equation (6), the visibility of a fragment depends on the transmission T_i at the pixel level, but this should be performed at the sub-pixel level according to Equation (4). Since the detail of the transmittance calculation for each sub-pixel is not included in (Maule et al. 2012), we define t_i^j within these triangle fragments according to Equation (2):

$$t_i^j = \prod_{k=1}^{i-1} (1 - o_k^j \alpha_k) \text{ and } t_1^j = 1.0. \tag{7}$$

Thus, O_i in Equation (6) is rewritten as:

$$O_i = \frac{1}{M} \sum_{j=1}^{M} o_i^j t_i^j. \tag{8}$$

O_i can be regarded as a continuous coverage influenced by the sub-pixel transmittance t_i^j through weighted averaging, differing from the M discrete values in Equation (4). Therefore, T_i is no longer required, as transmittance is now handled at the sub-pixel level to accommodate the needs of MSAA. As a result, we modify Equation (5) to:

$$C_{\text{tris}} = T_{\text{in}} \sum_{i=1}^{N} O_i \alpha_i c_i. \tag{9}$$

Notably, though T_i is dropped compared to Equation (5), it is still updated by Equation (6) for the subsequent blending. For example (see Figure 3b and 4b), given a background with C_{bg} and α_{bg}, the final color of the pixel is:

$$C_{\text{pixel}} = C_{\text{tris}} + T_{\text{bg}} C_{\text{bg}}, \tag{10}$$

with

$$T_{\text{bg}} = T_N (1 - \alpha_{\text{bg}}). \tag{11}$$

The main philosophy behind Equation (9) is treating the N triangles overlapping the same pixel as a single entity, so the updated transmittance T_N at the pixel level contributes

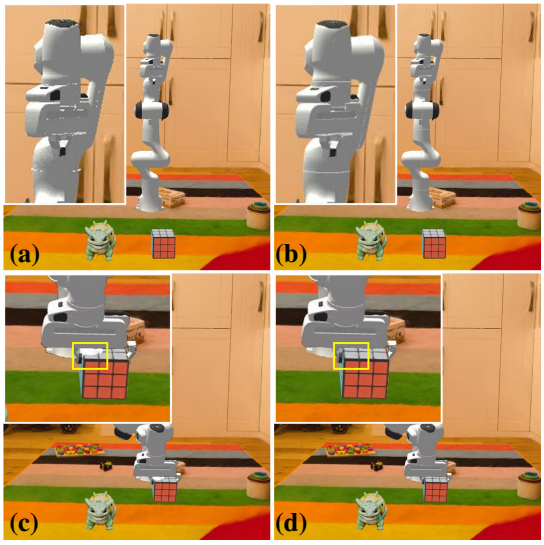


Figure 4: Visual improvements brought by our rasterization pipeline. (a): Direct α-blending; (b): Aliasing removed by combining MSAA with α-blending; (c): Color overflow caused by incorrect transmittance; (d): Our full method without artifacts.

correctly to Equation (11). However, this solution can not directly integrate with Gaussians. When there are Gaussians between triangles, the transmittance used in α-blending for triangles whose depth value is larger than Gaussians is too large (see the rightmost triangle in Figure 3c), as it does not consider the transmittance decay caused by the ray passing through Gaussians. This results in *color overflow* and manifests as white artifacts (see Figure 4c). To avoid *color overflow*, we propose a simple yet effective method, where only the depth-adjacent triangles are treated as a single entity (see the two left triangles in Figure 3d), rather than all triangles overlapping the pixel. During the unified rasterization of mesh and 3DGS, Equation (1) and Equation (9) are selected for Gaussian and triangular fragments, respectively, which is in a single-pass anti-aliased manner (see Figure 4d).

Deformation with Gaussian-Centric Binding

Binding Gaussians to Mesh. We advocate a Gaussian-centric perspective and employ ray casting to associate Gaussians with the mesh faces, which is training-free and robust to mesh topology. Since the cameras used in 3DGS training are inherently oriented toward the object, we cast rays from these cameras toward the center of a Gaussian. If a ray hits a face, we record the face index, the barycentric coordinate of the intersection point, and the distance from the intersection point to the Gaussian center. The Gaussian is then bound to the nearest candidate face. However, this is not an ideal choice, as the size mismatch between Gaussians and faces can be significant, potentially leading to inconsistent deformation magnitudes. Given that the cage or box deformation is robust (Tong et al. 2025b), we extend the Gaussian transformation to the average of the transformations of its BBX corners. To this end, we propose an extended strategy where each camera casts 8 rays toward the corners of a Gaussian's BBX. For each ray, we retain the face closest to the Gaussian. As a result, each Gaussian is ultimately bound to 8 faces. This method is implemented using OptiX (Parker et al. 2010). Since ray casting is fully parallelized, the binding process completes within a few seconds.

Deformation Transfer. As we bind 3DGS to the proxy mesh, any deformation method for meshes can be extended to 3DGS, such as physical simulation and modifiers from the graphics engine. Here, we briefly describe how deformation is transferred from the proxy mesh to 3DGS. Consider a Gaussian kernel G with its BBX B, parameterized by a mean vector μ and a covariance matrix Σ. For notational clarity, the subscript $i = \{1, 2, \cdots, 8\}$ in any quantity X_i^j refers to the index of a BBX corner, while the superscript $j = \{1, 2, 3\}$ specifies the vertex indices of the associated triangle face. Following GaussianMesh (Gao et al. 2024a), the ACAP (Gao et al. 2021) is employed to calculate the transformation of a vertex during manipulation. Specifically, the motion of a vertex consists of an offset Δ_i^j, a transformation matrix D_i^j that can be decomposed into a rotation matrix R_i^j and a shear matrix S_i^j via polar decomposition. Moreover, barycentric interpolation allows us to propagate the transformation from triangle vertices to any point within the triangle, which in our paper corresponds to the intersection point P_i with barycentric coordinates $\{u, v, w\}$. The transformation of P_i can be computed as follows:

$$\begin{aligned}
\Delta_i &= u\Delta_i^1 + v\Delta_i^2 + w\Delta_i^3 \\
&= u(V_i^{1\prime} - V_i^1) + v(V_i^{2\prime} - V_i^2) + w(V_i^{3\prime} - V_i^3), \\
R_i &= u\log(R_i^1) + v\log(R_i^2) + w\log(R_i^3), \text{ and} \\
S_i &= uS_i^1 + vS_i^2 + wS_i^3,
\end{aligned} \tag{12}$$

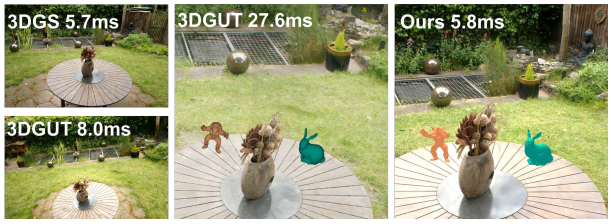
where V_i^j and $V_i^{j\prime}$ represent the position of the j-th vertex of a triangle before and after manipulation, respectively. Since the transformations of the i-th corner of B keep the same with P_i, they are subsequently transferred to G by averaging operations:

$$\begin{aligned}
&R' = \exp\left(\frac{1}{8}\sum_{i=1}^{8} R_i\right),\ S' = \frac{1}{8}\sum_{i=1}^{8} S_i,\ \Sigma' = R'S'\Sigma(R'S')^T, \\
&\mu' = \mu + \frac{1}{8}\sum_{i=1}^{8}\Delta_i, \text{ and } G'(x) = e^{-\frac{1}{2}(x-\mu')^T(\Sigma')^{-1}(x-\mu')},
\end{aligned} \tag{13}$$

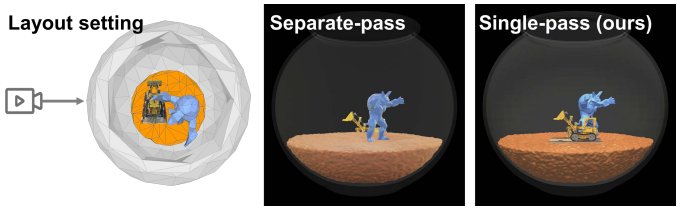
where $G'(x)$ is the updated Gaussian kernel. For more details, please see the references (Gao et al. 2024a, 2021).

Experiments

We comprehensively evaluate the capabilities of *UniMGS* through quantitative and qualitative experiments, including the performance of our unified rasterization, the improvement in deformation quality brought by the Gaussian-centric binding strategy, and two representative applications based on our unified framework. For our method, all Gaussian objects are reconstructed using 3DGS (Kerbl et al. 2023), while for others, we follow their official instructions to train



(a) Comparing with ray-based methods



(b) Comparing with separate-pass scheme

Figure 5: Evaluation against existing hybrid rendering studies. (a): Compared to ray-based methods, our method shows promising runtime performance while maintaining comparable visual effects. (b): Compared to separate-pass rasterization work, our method guarantees the correct spatial relationship because the colors of Gaussians and triangles are blended in a single pass.

Method	PSNR↑	SSIM↑	LPIPS↓	*Count*
UniMGS	**28.43**	**0.946**	**0.048**	275K
*UniMGS**	28.23	0.940	0.051	275K
GaussianMesh	27.21	0.925	0.079	989K
Frosting	25.42	0.912	0.059	2.16M
Mani-GS	23.76	0.842	0.123	2.47M

Table 1: Quantitative comparison of novel view synthesis on NeRF-Synthetic dataset.

them until convergence. Evaluations are based on the public dataset of NeRF-Synthetic (Mildenhall et al. 2020), Mip-NeRF360 (Barron et al. 2022), Hybrid-IBR (Prakash et al. 2021), and our self-collected data from Fab (Epic Games, Inc. 2025a). Due to space limitations, we only show a subset of the results. Additional examples are provided in the supplementary material.

Unified Rasterization Ability

Baselines. Existing studies (Xie et al. 2025; Wang et al. 2024; Xiao et al. 2024) on coupled rasterization of Gaussians and meshes all follow a separate-pass paradigm, but none of them are available. Therefore, we have to reproduce the separate-pass rendering scheme for an intuitive evaluation. In addition, 3DGUT (Wu et al. 2025) employs ray-based rendering for 3DGS and meshes. Implementation details are provided in the supplementary material.

Results. In Figure 5a, we designed a benchmark scene composed of Gaussians (*garden* from MipNeRF360 (Barron et al. 2022)) and low-poly meshes (shapes in orange and green), and recorded the average per-frame rendering time. With Gaussians only, 3DGUT takes 1.4× longer to render than 3DGS, consistent with its original findings. When meshes are added, our rendering speed remains stable, while 3DGUT slows down by 3.4×. This performance drop is attributed to 3DGUT's reliance on additional ray-tracing passes for meshes, while our approach employs a unified rasterization pipeline that handles both meshes and Gaussians in a single pass. Figure 5b depicts a scenario in which a *lego* represented by Gaussians is placed alongside mesh objects, enclosed by a transparent bowl. However, the separate-pass approach fails to handle occlusion in this case. In the separate-pass scheme, the *lego* and mesh images are composited pixel-wise based on depth. However, due to the nested spatial relationship, the *lego* is impossible to appear inside the bowl. The key reason is that correct visibility can only be ensured within the same representation, while depth maps alone are insufficient to guarantee the correctness of subsequent color composition. In contrast, our single-pass method achieves true unified rendering by computing color at the fragment level.

Deformation Performance

This experiment evaluates the deformation performance of mesh-centric and our Gaussian-centric binding strategies across different animation algorithms and proxy mesh sources. Visual artifacts serve as a primary indicator. Several advanced deformation representations have been proposed by GaussianMesh (Gao et al. 2024a), Mani-GS (Gao et al. 2024b), and Frosting (Guédon and Lepetit 2024a), which are mesh-centric methods. For our method, we refer to the direct bind of a Gaussian to a single face as *UniMGS**, and the BBX-based association to multiple faces as *UniMGS*.

Quantitative Comparison on NeRF-Synthetic Dataset. In the NeRF-Synthetic dataset (Mildenhall et al. 2020), the artist-created Ground Truth (GT) meshes are employed as proxies, whose triangle face count ranges from 60K to 1,200K. Following PhysGaussian (Xie et al. 2024), we perform deformation with the Simple Deform Modifier in Blender (Blender Foundation 2025) and evaluate the results using PSNR, SSIM, and LPIPS. These metrics are computed by comparing the novel view renderings of the deformed mesh and 3DGS. In addition, the average Gaussian count is reported as *Count*.

As reported in Table 1, our method outperforms others in all quantitative metrics, where the corresponding visualizations are provided in the supplementary material. In GaussianMesh, faces and Gaussians could be subdivided together during training; Frosting introduces an additional frosting layer composed of Gaussians; and Mani-GS assigns multiple Gaussians to each face. These mesh-centric strategies must bind Gaussians to the mesh before training, guided by geometric constraints from the proxy mesh. While beneficial for static 3DGS optimization, such constraints may impair deformation quality and produce artifacts. Instead, our Gaussian-centric strategy directly links the trained 3DGS to

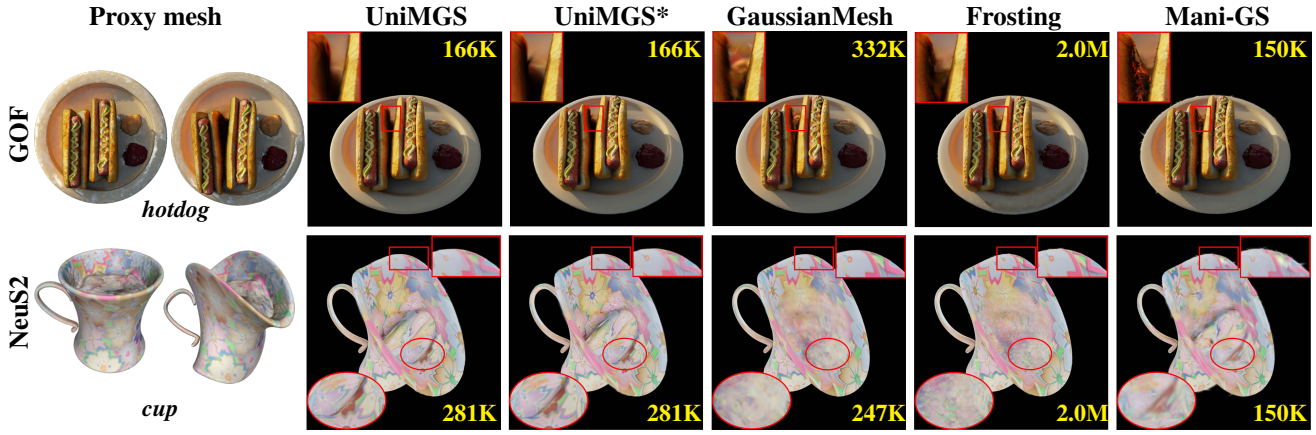


Figure 6: Visual comparison of deformation. We visualize textured proxy meshes before and after deformation, obtained from two advanced reconstruction methods (NeuS2 and GOF), where the number of Gaussians is marked in yellow. All proxy meshes are simplified to 50K faces. The baselines (GaussianMesh, Frosting, and Mani-GS) suffer from severe artifacts under large deformations, primarily due to their face-centric binding strategies, which are inherently sensitive to the quality of the proxy mesh topology.

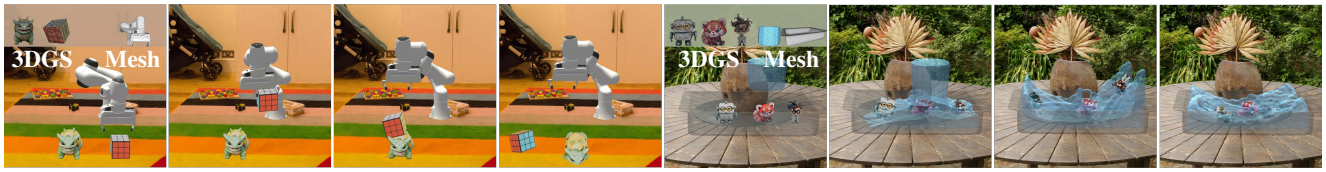


Figure 7: Applications in embodied AI and fluid simulation. These two scenarios involve complex occlusions, rich interactions, and diverse appearances (including both transparent and opaque objects), which are all faithfully presented. The backgrounds are reconstructed by 3DGS (Kerbl et al. 2023).

the mesh, thus keeping the original 3DGS data unchanged, while preserving promising results.

Robustness to Mesh Quality. Since high-quality meshes are not always available, the proxy meshes in this experiment are obtained from NeuS2 (Wang et al. 2023) (recommended by GaussianMesh and Mani-GS) and GOF (Yu, Sattler, and Geiger 2024). Considering computational cost, we follow GaussianMesh (Gao et al. 2024a) and simplify all proxy meshes to 50K faces using the decimation method (Garland and Heckbert 1997; Cignoni et al. 2008). Physical simulation (Macklin, Müller, and Chentanez 2016) is imposed on *hotdog*, whereas Blender modifiers are employed for *cup*. Figure 6 illustrates the proxy meshes before and after deformation, along with the resulting deformed 3DGS. GOF is an advanced method for extracting textured meshes from 3DGS, while NeuS2 relies on implicit neural fields and consistently generates watertight meshes. However, NeuS2 exhibits a failure case that erroneously closes the opening of the *cup*. Under such an unfavorable condition, existing mesh-centric methods perform poorly in Gaussian optimization, as Gaussians are constrained near the proxy mesh surface. In contrast, our Gaussian-centric approach spatially links Gaussians to the mesh without imposing a causal relationship, thus offering greater tolerance even when the proxy mesh is of poor quality or structurally flawed. Disregarding mesh quality, GaussianMesh also adopts ACAP for deformation transfer, but suffers from distortion and blurring, further highlighting the advantage of our Gaussian-centric binding strategy. Compared to *UniMGS**, the introduction of BBX in *UniMGS* improves visual performance.

Applications Based on UniMGS

UniMGS unifies mesh and 3DGS in both rasterization and deformation, laying the foundation for downstream tasks involving hybrid representations. Figure 7 shows two representative cases to demonstrate the application potential of UniMGS in embodied AI and fluid simulation. Implementation details are provided in the supplementary material.

Conclusion

In this paper, we first address the underexplored problem of unified rasterization of 3DGS and mesh by blending the colors of both triangle and Gaussian fragments in a single-pass manner, ensuring accurate color computation and occlusion handling. Second, we propose a novel Gaussian-centric binding strategy to enhance the visual performance of 3DGS driven by proxy meshes. Finally, comprehensive experiments validate the superiority of *UniMGS* in rendering and manipulation, while the application cases further emphasize the practical significance.

References

Barron, J. T.; Mildenhall, B.; Verbin, D.; Srinivasan, P. P.; and Hedman, P. 2022. Mip-NeRF 360: Unbounded Anti-Aliased Neural Radiance Fields. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 5470–5479.

Blender Foundation. 2025. Blender 4.4 Manual. https://www.blender.org/. [Online; accessed 25-April-2025].

Byrski, K.; Mazur, M.; Tabor, J.; Dziarmaga, T.; Kadziolka, M.; Baran, D.; and Spurek, P. 2025. RaySplats: Ray Tracing Based Gaussian Splatting. *arXiv preprint arXiv:2501.19196*.

Cignoni, P.; Callieri, M.; Corsini, M.; Dellepiane, M.; Ganovelli, F.; and Ranzuglia, G. 2008. MeshLab: an Open-Source Mesh Processing Tool. In Scarano, V.; Chiara, R. D.; and Erra, U., eds., *Eurographics Italian Chapter Conference*. The Eurographics Association. ISBN 978-3-905673-68-5.

de Vries, J. 2020. *Learn OpenGL: Learn Modern OpenGL Graphics Programming in a Step-by-Step Fashion (Inbook-text-in-chap)*, chapter: Part IV - Advanced OpenGL Anti Aliasing, 264–271. Kendall & Welling. ISBN 978-90-90-33256-7.

Epic Games, Inc. 2025a. Fab. https://www.fab.com/. [Online; accessed 18-May-2025].

Epic Games, Inc. 2025b. Unreal Engine 5.6 Documentation: Anti-Aliasing and Upscaling. https://dev.epicgames.com. [Online; accessed 20-June-2025].

Gao, L.; Lai, Y.-K.; Yang, J.; Zhang, L.-X.; Xia, S.; and Kobbelt, L. 2021. Sparse Data Driven Mesh Deformation. *IEEE Transactions on Visualization and Computer Graphics*, 27(3): 2085–2100.

Gao, L.; Yang, J.; Zhang, B.-T.; Sun, J.-M.; Yuan, Y.-J.; Fu, H.; and Lai, Y.-K. 2024a. Real-time Large-Scale Deformation of Gaussian Splatting. *ACM Trans. Graph.*, 43(6): 1–17.

Gao, X.; Li, X.; Zhuang, Y.; Zhang, Q.; Hu, W.; Zhang, C.; Yao, Y.; Shan, Y.; and Quan, L. 2024b. Mani-GS: Gaussian Splatting Manipulation with Triangular Mesh. *arXiv preprint arXiv:2405.17811*.

Garland, M.; and Heckbert, P. S. 1997. Surface Simplification Using Quadric Error Metrics. In *Proceedings of the 24th Annual Conference on Computer Graphics and Interactive Techniques*, 209–216.

Guédon, A.; and Lepetit, V. 2024a. Gaussian Frosting: Editable Complex Radiance Fields with Real-Time Rendering. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 413–430. Springer.

Guédon, A.; and Lepetit, V. 2024b. SuGaR: Surface-Aligned Gaussian Splatting for Efficient 3D Mesh Reconstruction and High-Quality Mesh Rendering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 5354–5363.

Guo, Y.-C.; Cao, Y.-P.; Wang, C.; He, Y.; Shan, Y.; and Zhang, S.-H. 2023. VMesh: Hybrid Volume-Mesh Representation for Efficient View Synthesis. In *SIGGRAPH Asia 2023 Conference Papers*, 1–11.

Huang, J.; and Yu, H. 2024. GSDeformer: Direct Cage-Based Deformation for 3D Gaussian Splatting. *arXiv preprint arXiv:2405.15491*.

Huang, Y.-H.; Sun, Y.-T.; Yang, Z.; Lyu, X.; Cao, Y.-P.; and Qi, X. 2024. SC-GS: Sparse-Controlled Gaussian Splatting for Editable Dynamic Scenes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 4220–4330.

Hughes, J. F.; van Dam, A.; McGuire, M.; Sklar, D. F.; Foley, J. D.; Feiner, S. K.; and Akeley, K. 2013. *Computer Graphics Principles and Practice (Inbook-text-in-chap)*, chapter: Ray Casting and Raterization, 387–393. Upper Saddle River, NJ, USA: Addison-Wesley Professional. ISBN 978-0-321-39952-6.

Kazhdan, M.; Bolitho, M.; and Hoppe, H. 2006. Poisson Surface Reconstruction. In *Proceedings of the Fourth Eurographics Symposium on Geometry Processing*, volume 7.

Kerbl, B.; Kopanas, G.; Leimkühler, T.; and Drettakis, G. 2023. 3D Gaussian Splatting for Real-Time Radiance Field Rendering. *ACM Trans. Graph.*, 42(4): 139–1.

Li, Z.; Chen, Y.; and Liu, P. 2024. DreamMesh4D: Video-to-4D Generation with Sparse-Controlled Gaussian-Mesh Hybrid Representation. *Advances in Neural Information Processing Systems*, 37: 21377–21400.

Macklin, M.; Müller, M.; and Chentanez, N. 2016. XPBD: Position-Based Simulation of Compliant Constrained Dynamics. In *Proceedings of the 9th International Conference on Motion in Games*, 49–54.

Maule, M.; Comba, J. L.; Torchelsen, R.; and Bastos, R. 2012. Transparency and Anti-Aliasing Techniques for Real-Time Rendering. In *2012 25th SIBGRAPI Conference on Graphics, Patterns and Images Tutorials*, 50–59. IEEE.

Mildenhall, B.; Srinivasan, P. P.; Tancik, M.; Barron, J. T.; Ramamoorthi, R.; and Ng, R. 2020. NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 405–421. Springer.

Moenne-Loccoz, N.; Mirzaei, A.; Perel, O.; de Lutio, R.; Esturo, J. M.; State, G.; Fidler, S.; Sharp, N.; and Gojcic, Z. 2024. 3D Gaussian Ray Tracing: Fast Tracing of Particle Scenes. *ACM Transactions on Graphics (SIGGRAPH Asia)*.

Parker, S. G.; Bigler, J.; Dietrich, A.; Friedrich, H.; Hoberock, J.; Luebke, D.; McAllister, D.; McGuire, M.; Morley, K.; Robison, A.; and Stich, M. 2010. OptiX: A General Purpose Ray Tracing Engine. *ACM Trans. Graph.*, 29(4).

Prakash, S.; Leimkühler, T.; Rodriguez, S.; and Drettakis, G. 2021. Hybrid Image-Based Rendering for Free-View Synthesis. *Proceedings of the ACM on Computer Graphics and Interactive Techniques*, 4(1): 1–20.

Qiao, Y.-L.; Gao, A.; Xu, Y.; Feng, Y.; Huang, J.-B.; and Lin, M. C. 2023. Dynamic Mesh-Aware Radiance Fields. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 385–396.

Tong, X.; Shao, T.; Weng, Y.; Yang, Y.; and Zhou, K. 2025a. As-Rigid-As-Possible Deformation of Gaussian Radiance Fields. *IEEE Transactions on Visualization and Computer Graphics*.

Tong, Y.; Tian, R.; Han, X.; Liu, D.; Yu, F.; and Zhang, Y. 2025b. CAGE-GS: High-Fidelity Cage Based 3D Gaussian Splatting Deformation. *arXiv preprint arXiv:2504.12800*.

Waczyńska, J.; Borycki, P.; Tadeja, S.; Tabor, J.; and Spurek, P. 2024. GAMES: Mesh-Based Adapting and Modification of Gaussian Splatting. *arXiv preprint arXiv:2402.01459*.

Wang, C.; Kang, D.; Sun, H.-Y.; Qian, S.-H.; Wang, Z.-X.; Bao, L.; and Zhang, S.-H. 2024. Mega: Hybrid Mesh-Gaussian Head Avatar for High-Fidelity Rendering and Head Editing. *arXiv preprint arXiv:2404.19026*.

Wang, Y.; Han, Q.; Habermann, M.; Daniilidis, K.; Theobalt, C.; and Liu, L. 2023. Neus2: Fast Learning of Neural Implicit Surfaces for Multi-View Reconstruction. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 3295–3306.

Wu, G.; Yi, T.; Fang, J.; Xie, L.; Zhang, X.; Wei, W.; Liu, W.; Tian, Q.; and Wang, X. 2024. 4D Gaussian Splatting for Real-Time Dynamic Scene Rendering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 20310–20320.

Wu, Q.; Martinez Esturo, J.; Mirzaei, A.; Moenne-Loccoz, N.; and Gojcic, Z. 2025. 3DGUT: Enabling Distorted Cameras and Secondary Rays in Gaussian Splatting. *Conference on Computer Vision and Pattern Recognition (CVPR)*.

Xia, H.; Lin, Z.-H.; Ma, W.-C.; and Wang, S. 2024. Video2Game: Real-Time Interactive Realistic and Browser-Compatible Environment from a Single Video. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 4578–4588.

Xiao, Y.; Wang, X.; Li, J.; Cai, H.; Fan, Y.; Xue, N.; Yang, M.; Shen, Y.; and Gao, S. 2024. Bridging 3D Gaussian and Mesh for Freeview Video Rendering. *arXiv preprint arXiv:2403.11453*.

Xie, T.; Zong, Z.; Qiu, Y.; Li, X.; Feng, Y.; Yang, Y.; and Jiang, C. 2024. PhysGaussian: Physics-Integrated 3D Gaussians for Generative Dynamics. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 4389–4398.

Xie, Z.; Liu, Z.; Peng, Z.; Wu, W.; and Zhou, B. 2025. Vid2Sim: Realistic and Interactive Simulation from Video for Urban Navigation. *arXiv preprint arXiv:2501.06693*.

Ye, K.; Wu, H.; Tong, X.; and Zhou, K. 2024. A Real-time Method for Inserting Virtual Objects into Neural Radiance Fields. *IEEE Transactions on Visualization and Computer Graphics*.

Yu, Z.; Sattler, T.; and Geiger, A. 2024. Gaussian Opacity Fields: Efficient Adaptive Surface Reconstruction in Unbounded Scenes. *ACM Trans. Graph.*, 43(6): 1–13.

Zhong, L.; Yu, H.-X.; Wu, J.; and Li, Y. 2024. Reconstruction and Simulation of Elastic Objects with Spring-Mass 3D Gaussians. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 407–423. Springer.

Zwicker, M.; Pfister, H.; Van Baar, J.; and Gross, M. 2001. EWA Volume Splatting. In *Proceedings Visualization, 2001. VIS'01.*, 29–538. IEEE.