Mani-GS: Gaussian Splatting Manipulation with Triangular Mesh

Xiangjun Gao[1] Xiaoyu Li[2] † Yiyu Zhuang[3] Qi Zhang[2] Wenbo Hu[2] Chaopeng Zhang[2]
Yao Yao[3] † Ying Shan[2] Long Quan[1]

[1]The Hong Kong University of Science and Technology [2]Tencent [3]Nanjing University

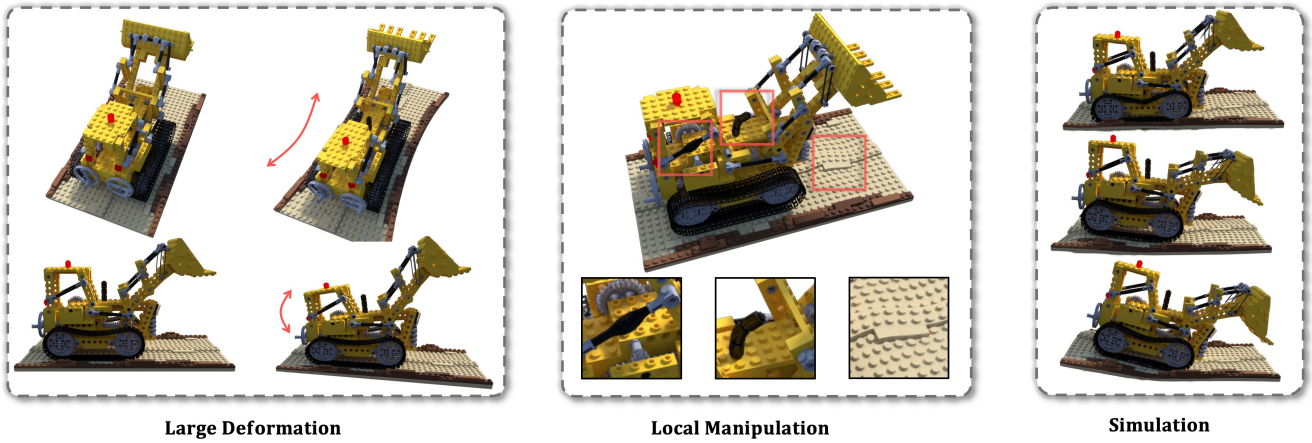

Figure 1. Our proposed approach allows for 3DGS manipulation, including *large deformation*, *local manipulation*, and even *physical simulation* (such as soft body), while maintaining high-quality rendering.

Abstract

Neural 3D representations, such as Neural Radiation Fields (NeRF), excel at producing photorealistic rendering results but lack the flexibility for manipulation and editing which is crucial for content creation. However, manipulating NeRF is not highly controllable and requires a long training and inference time. With the emergence of 3D Gaussian Splatting (3DGS), extremely high-fidelity novel view synthesis can be achieved using an explicit point-based 3D representation with much faster training and rendering speed. However, there is still a lack of effective means to manipulate 3DGS freely while maintaining rendering quality. In this work, we aim to tackle the challenge of achieving manipulable photo-realistic rendering. We propose to utilize a triangular mesh to manipulate 3DGS directly with self-adaptation. This approach reduces the need to design various algorithms for different types of 3DGS manipulation. By utilizing a triangle shape-aware Gaussian binding and adapting method, we can achieve 3DGS manipulation and preserve high-fidelity rendering. In addition, our method is also effective with inaccurate meshes extracted from 3DGS. Experiments demonstrate our method's effectiveness and superiority over baseline approaches.

1. Introduction

Editing 3D content is essential for content creation and has various applications in movies, gaming, and virtual/augmented reality. 3D model editing enables users to create and modify models flexibly, thereby enhancing production efficiency. The traditional pipeline for modeling and editing a 3D asset with photo-realistic rendering involves a process with geometry modeling, texturing, UV mapping, lighting, and rendering, which is a tedious and time-consuming flow requiring lots of manual work.

Over the past few years, the neural radiance field (NeRF) [26] has been widely studied due to its high capability and simple reconstruction process in 3D representation. However, the implicit representation poses challenges for editing. To address this, some methods are proposed to edit this implicit neural radiance field [26]. NeRF-Editing [50] is the first to utilize the triangular mesh to help edit the implicit radiance field. Volume rendering is conducted in the deformed space to render the deformed object by editing the

†Corresponding author: Xiaoyu Li and Yao Yao.

triangular mesh. The sampling points in the deformed space are mapped to canonical space based on the constructed tetrahedra grid. [15, 23] present similar ideas by employing tetrahedra to deform sampling points and achieve editable rendering. In addition, NeuMesh [44] and SERF [54] define the neural radiance field by associating each mesh vertex with radiance and geometry features. Then they conduct volume rendering like Point-NeRF [42] in deformed space without backward mapping. However, these editing methods based on implicit neural radiance fields still suffer from inconvenient manipulation, suboptimal rendering results, and long training and rendering times.

Recently, 3DGS [19] has gained significant attention in differential rendering due to its high-fidelity and fast rendering proficiency. However, despite being an explicit 3D representation, it still lacks an effective method for manipulating 3DGS while maintaining high-quality rendering. SuGaR [10], the work most closely related to our objective, develops a novel algorithm to extract a triangular mesh from 3DGS. Although their main goal is not to facilitate editable photorealistic rendering, they bind the 3DGS to the extracted mesh, enabling model animation.

This paper proposes a method that enables 3DGS manipulation, achieving high-quality and photo-realistic rendering. Our key insight is to manipulate 3DGS using a triangular mesh as the proxy, which allows for direct transfer of mesh manipulation to 3DGS with 3DGS self-adaptation. With our methods, we can achieve *large deformation*, *local manipulation*, and *soft body simulaton* with high-quality results as shown in Fig 1, which also avoid the need to design various algorithms for different types of manipulation.

To achieve controllable 3DGS manipulation through mesh, an intuitive approach is to bind the GS to lay perfectly on the triangle and enforce the GS to be thin enough. After mesh manipulation, the GS will automatically adapt its rotation and position with the attached triangle, as employed in SuGaR [10]. However, SuGaR heavily relies on the accuracy of mesh geometry, inheriting the defects of mesh rendering. Specifically, for inaccurate parts, SuGaR cannot inpaint the missing parts or remove the redundant parts during rendering.

Adding an offset to the position of attached Gaussians during the reconstruction may seem like a reasonable solution to compensate for mesh inaccuracy. However, this fixed offset cannot be generalized well to the deformed space after manipulation. Our proposed solution is to define a local coordinate system for each triangle, which we refer to as *local triangle space*. We then bind Gaussians to each triangle and optimize the Gaussian attributes, including rotation, position, and scaling, in the attached *local triangle space*.

During mesh manipulation, the attributes in the local triangle space remain unchanged, while the global Gaussian position, scaling, and rotation will be self-adaptively adjusted according to our proposed formula. As a result, our proposed approach enables us to manipulate 3DGS using a triangular mesh while maintaining rendering quality. Since our Gaussians are set to be free outside the triangle, we can also support high-fidelity manipulation even when the Gaussians are bound to an inaccurate mesh, exhibiting a high tolerance for mesh accuracy.

The contributions of our paper are listed as follows:

- We propose a 3DGS manipulation method that can effectively transfer the triangular mesh manipulation to 3DGS and maintain high-quality rendering.
- We introduce a triangle shape aware Gaussian binding strategy with self-adaption, which has a high tolerance for mesh accuracy.
- We evaluate our method and achieve state of the art results, also supporting various 3DGS manipulations including *large deformation*, *local manipulation*, and *soft body simulation*.

2. Related Work

2.1. NeRF Editing

Recently, NeRF [26] has garnered significant attention due to its high-quality and photo-realistic rendering results for novel view synthesis. NeRF represents the scene as a continuous function that maps a spatial location and viewing direction to a volume density and color, which is parameterized by a multilayer perceptron (MLP). Owing to the implicit representation that encodes the scene within the network parameters, editing and deforming the geometry of the NeRF scene explicitly like mesh can be challenging. To enable user editing of NeRF, [24] introduce editing conditional radiance fields trained on a shape category. However, it only supports basic editing, such as removing/adding object parts or shape transfer. CLIP-NeRF [34] achieves NeRF editing with text or images by leveraging CLIP model [28] but still can not edit the geometry locally. Some other work [2, 37, 40, 51] edit the NeRF in texture level which is not the focus of this paper.

To edit and deform NeRF locally, [15, 23, 50] construct a tetrahedra grid based on the underlying 3D shape. After explicitly deforming the tetrahedra into the posed space for editing, the sampled 3D points are mapped from the posed space to the canonical space through the unaltered tetrahedron, which means the canonical position can be calculated from the shared barycentric coordinate for both deformed and canonical tetrahedron. The density and radiance in the posed space can be calculated for the mapped sampling points in canonical space. On the other hand, [35, 44, 54] employ mesh as the guidance for deformation. NeuMesh [44] presents a novel representation to encode neural implicit field on a mesh-based scaffold for geometry and texture editing. [35] achieves the manipulation of both

the geometry and color of neural implicit fields through differentiable colored meshes. Furthermore, there are several methods [3, 8, 25, 30, 39, 52, 53, 56, 57] that particularly focus on editing NeRF for avatar. In this work, we introduce an editing method based on 3DGS for general objects.

2.2. Mesh-based NeRF Rendering

NeRFs have shown impressive rendering results, however, rendering one pixel using NeRF necessitates a volumetric rendering algorithm that involves inferring MLP hundreds of times to estimate their radiance and density. To accelerate NeRF rendering, several methods [4, 9, 12, 29, 32, 46–48] have been proposed to combine NeRF representation with mesh reconstruction. By converting this implicit representation to an explicit mesh, these methods may also facilitate applications like editing. In particular, MobileNeRF [4] represents NeRF as a collection of polygons with deep feature textures, which can be rendered using the classic polygon rasterization pipeline. Additionally, NeRFMeshing [29] distills the reconstructed NeRF into a signed surface approximation network to extract 3D mesh and shows the simulation results for editing. However, the editing results of these methods heavily rely on the accuracy of the extracted mesh. Artifacts present in the mesh will directly influence the editing results. In contrast, our method demonstrates robustness to the reconstructed mesh and can still yield promising results even with the inaccurate mesh.

2.3. Gaussian Splatting Editing, Simulation and Animation

3D Gaussian Splatting [19] presents an innovative 3D Gaussian scene representation, accompanied by a differentiable renderer that attains real-time rendering of radiance fields while maintaining high quality. Initially, 3D Gaussian Splatting focuses solely on static scenes, which has been extended to model dynamic scenes [14, 22, 38, 45, 55], human avatars [1, 13, 20, 27, 43, 49, 58], Gaussian Splatting simulation and animation [5, 10, 17, 41]. Specifically, SuGaR [10] proposes a method to extract meshes from 3D Gaussian Splatting with additional regularization, in which they bind 3DGS on extracted mesh and animated with Gaussian Splatting rendering. GSP [5] incorporates physically-based fluid dynamics in 3DGS and PhysGaussian [41] introduces a unified simulation-rendering pipeline that generates physics-based dynamics with photorealistic renderings. VR-GS [17] achieves interactive physics-based editing in Virtual Reality. In this work, we also introduce a 3DGS manipulation method that binds Gaussian Splatting to the mesh, achieving state-of-the-art results.

Several concurrent works have emerged in parallel with ours. In particular, GaMeS [33] constrains the 3DGS should be a flat ellipse and lay on the surface tightly, which means the position of 3DGS is inside the triangle and rotation is the same with triangle rotation; Mesh-GS [7] permits an offset along the normal direction but the rotation and scaling are fixed after manipulation. Our model allows 3DGS to move out of the triangle and maintain the relative position and rotation after manipulation. And we can achieve high-quality rendering without the need for accurate mesh. Following SuGaR, Guedon et al. [11] proposed to represent the scene using a multi-layer mesh, while our method requires only a single-layer mesh.

3. Method

Recently, due to its exceptional high-fidelity rendering capabilities and fast rendering speed, 3DGS [19] has emerged as a popular 3D representation in the differential rendering field. However, despite being an explicit 3D representation, it still lacks a way to manipulate this 3D representation for editing while maintaining high-quality rendering after the manipulation. In this work, giving multi-view RGB images of an object as input, we introduce a method for object manipulation that can achieve photo-realistic editable rendering by employing 3DGS.

The pipeline of our method is illustrated in Figure 2 and consists of three main stages. First, we extract a mesh from 3D Gaussian Splatting (3DGS) or a neural surface field for subsequent 3D Gaussian binding (Sec. 3.2). Next, we devise a novel Mesh-Gaussian binding method dedicated to manipulating 3DGS while maintaining photo-realistic rendering quality (Sec. 3.3). Finally, we describe the types of Gaussian manipulation we support(Sec. 3.4).

3.1. Preliminary

Thanks to its superior capability for high-fidelity and rapid rendering, 3D Gaussian Splatting (3DGS) [19] has recently emerged as a popular 3D representation in differential rendering. 3DGS utilizes explicit 3D Gaussians as its primary rendering primitives. A 3D Gaussian point is mathematically defined as:

$$G(\boldsymbol{x}) = exp(-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu})^\top \Sigma^{-1}(\boldsymbol{x}-\boldsymbol{\mu})). \quad (1)$$

Each 3D Gaussian point is characterized by a 3D mean position coordinate $\boldsymbol{\mu}$ and a covariance matrix Σ. Additionally, each Gaussian has an opacity $\boldsymbol{o}$ and a view-dependent color $\boldsymbol{c}$ represented by a set of spherical harmonics (SH). To ensure that the covariance matrix Σ retains its meaningful interpretation, it is parameterized as a unit quaternion $\boldsymbol{q}$ and a 3D scaling vector $\boldsymbol{s}$, defined as $\Sigma = \boldsymbol{RSS}^\top \boldsymbol{R}^\top$.

To render an image from a specific viewpoint, 3D Gaussians are projected onto the image plane, resulting in 2D Gaussians. The 2D covariance matrix is approximated as:

$$\Sigma' = \boldsymbol{JW}\Sigma\boldsymbol{W}^\top\boldsymbol{J}^\top, \quad (2)$$

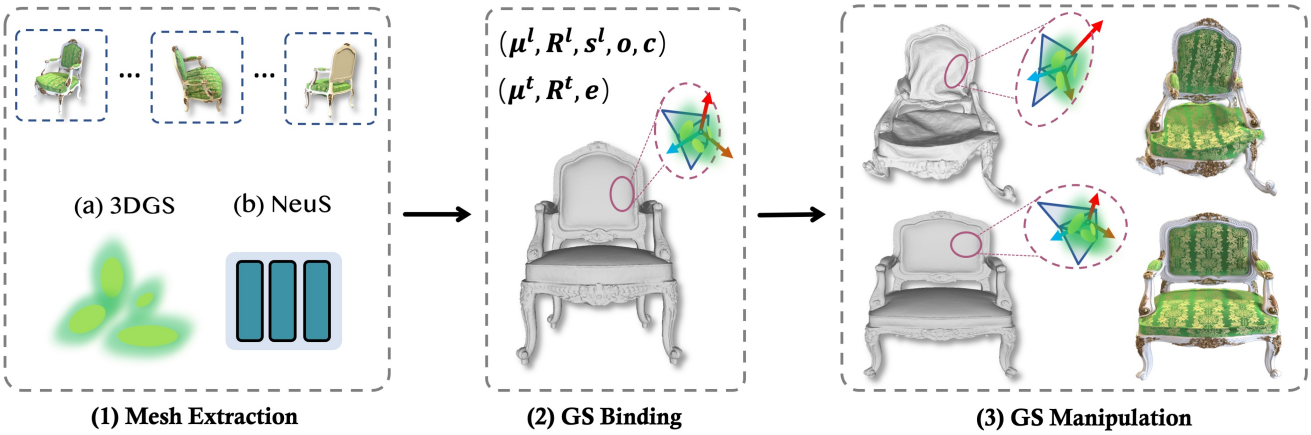

Figure 2. Overview of our method.(1) Firstly, we extract a triangular mesh from 3DGS [19] or a neural surface field(NeuS [36]). (2) Next, we bind N Gaussians to each triangle in the local triangle space, and optimize the local Gaussian attributes $(\boldsymbol{\mu^l}, \boldsymbol{R^l}, \boldsymbol{s^l}, \boldsymbol{o}, \boldsymbol{c})$. The triangle attributes $(\boldsymbol{\mu^t}, \boldsymbol{R^t}, \boldsymbol{e})$ are calculated based on the triangle vertices. (3) Finally, we manipulate 3DGS by transferring the mesh manipulation directly, thus achieving manipulable rendering.

where $\boldsymbol{W}$ and $\boldsymbol{J}$ denote the viewing transformation and the Jacobian of the affine approximation of perspective projection transformation [59], respectively. The 2D means are calculated through the projection matrix. After this, the pixel color is composited through the alpha blending of N ordered 2D Gaussians:

$$\mathcal{C} = \sum_{i \in N} T_i \alpha_i \boldsymbol{c}_i \text{ with } \quad T_i = \prod_{j=1}^{i-1} (1 - \alpha_i). \tag{3}$$

Here, α is obtained by multiplying the opacity $\boldsymbol{o}$ with the 2D covariance's probability computed from Σ' and pixel coordinate on the image space.

3.2. Mesh Extraction

Our method can achieve high-quality editing using guided meshes obtained from various methods. In this section, we investigate different mesh extraction and reconstruction techniques with different mesh accuracy and processing time to guide the 3D Gaussians in our approach.

Marching Cube for Gaussian Splatting. In DreamGaussian [31], the method attempts to summarize the alpha values of neighboring Gaussian points as the composite density value of marching cube sampling points. The mesh extracted using this method often ignores the thin and small structures. With our Mesh-Gaussian binding strategy, we can achieve high-fidelity rendering with the inaccurate mesh and support smooth Gaussian manipulation.

Screened Poisson Reconstruction. 3D Gaussian Splatting could be considered a type of point cloud, making it intuitive to extract the mesh using the Poisson-reconstruction algorithm. However, the 3D Gaussians do not have normal vectors for reconstruction. Inspired by recent 3DGS inverse rendering methods [6, 21], we allocate an additional gaussian attribute, normal $\boldsymbol{n}$, for 3D Gaussians, which is supervised by the pseudo normal derived from depth map. After training 3DGS with normal attributes, we can extract the mesh using the Screened poisson surface reconstruction [18] algorithm.

Neural Implicit Surfaces. In this work, we also try to extract high-quality surfaces from the implicit representation utilizing the method proposed in NeuS [36]. NeuS mesh has a large number of triangles, which negatively affects both training and inference speeds. We utilize mesh decimation techniques to reduce the count of triangles to approximately 300K.

3.3. Binding Gaussian Splatting on Mesh

Owing to the exceptional proficiency in high-fidelity and fast rendering, 3DGS has gained significant attention in differential rendering. However, despite being an explicit 3D representation, it currently lacks a method for effectively manipulating 3DGS while preserving high-quality rendering simultaneously. Mesh editing techniques, such as large-scale deformation, localized manipulation, and simulation, have been widely acknowledged and extensively researched for many years. Our primary objective is to associate the 3DGS with mesh triangles, enabling the manipulation of 3DGS and its rendering results following mesh editing.

Given a reconstructed or extracted triangular mesh $\boldsymbol{T}$ with K vertices $\{\boldsymbol{v}_i\}_{i=1}^{K}$ and M triangles $\{\boldsymbol{f}_i\}_{i=1}^{M}$, the goal of our method is to construct a 3DGS model bound to mesh triangles and optimize each Gaussian attribute $\{\boldsymbol{\mu}_i, \boldsymbol{q}_i, \boldsymbol{s}_i, o_i, \boldsymbol{c}_i\}$. To simplify the notation, we will omit the subscript in subsequent sections.

For each triangle $\boldsymbol{f}$ in the given mesh $\boldsymbol{T}$, which is composed of three vertices $(\boldsymbol{v_1}, \boldsymbol{v_2}, \boldsymbol{v_3})$, we initialize $N = 3$ Gaussians on this triangle. To be specific, the mean po-

sition $\boldsymbol{\mu}$ of initialized Gaussians is formulated as $\boldsymbol{\mu} = (w_1\boldsymbol{v_1} + w_2\boldsymbol{v_2} + w_3\boldsymbol{v_3})$, $\boldsymbol{w} = (w_1, w_2, w_3)$ is the predefined barycentric coordinate of each Gaussians attached on the triangle. And $\boldsymbol{w}$ satisfy $(w_1 + w_2 + w_3) = 1$.

Gaussians on Mesh. To achieve controllable 3DGS manipulation through the mesh, an intuitive way is to perfectly attach 3DGS to the triangle, as shown in the SuGaR [10]. With the rotation matrix denoted as $\boldsymbol{R} = \{\boldsymbol{r_1}, \boldsymbol{r_2}, \boldsymbol{r_3}\}$ and the scaling vector represented by $\boldsymbol{s} = (s_1, s_2, s_3)$, SuGaR train 3DGS with a flat Gaussian distribution on the mesh by setting $s_1 = \epsilon$, where ϵ is close to zero. $\boldsymbol{r_1}$ is defined by the normal vector $\boldsymbol{n}$ of the attached triangle. The Gaussians have only 2 learnable scaling factors (s_2, s_3) instead of 3, and only 1 learnable 2D rotation rather than a quaternion, to keep the Gaussians flat and aligned with the mesh triangles. This binding strategy is heavily dependent on mesh accuracy, which limits the flexibility of 3DGS in modeling complex object rendering. When the mesh accuracy is low, it cannot compensate for missing or redundant parts of geometry. Conversely, when the mesh accuracy is high but still deviates from the ground truth, it tends to produce a blurred effect due to the view inconsistency between the inaccurate mesh and multiview images. Overall, this binding strategy inherits the shortcomings of mesh rendering.

Gaussians on Mesh with Offset. To compensate for the inaccuracy of the extracted mesh, it would be better to add an offset $\Delta\boldsymbol{\mu}$ to the Gaussians 3D mean $\boldsymbol{\mu}$, which enables the Gaussians to move out of the attached triangle $\boldsymbol{f}$. Although it can improve the rendering quality of the reconstructed static object, this offset field is not generalized. It would result in noisy and unexpected rendering distortion in the manipulated object due to the mismatched relative position between 3DGS.

Triangle Shape Aware Gaussian Binding and Adaption. To preserve the high-fidelity rendering results after manipulation, the key lies in maintaining the local rigidity and preserving the relative location between Gaussians, both for 3D means and rotations. Our key insight is to define a local coordinate system in each triangle space.

The first axis direction of triangle space is defined as the direction of the first edge. The second axis direction of triangle space is defined as the triangle's normal direction. The third axis direction of triangle space is defined as the cross product of the first and second axis. Then the triangle coordinate system rotation can be formulated as:

$$\boldsymbol{R^t} = [\boldsymbol{r_1^t}, \boldsymbol{r_2^t}, \boldsymbol{r_3^t}] = [\frac{(\boldsymbol{v_2} - \boldsymbol{v_1})}{\|\boldsymbol{v_2} - \boldsymbol{v_1}\|}, \boldsymbol{n^t}, \frac{(\boldsymbol{v_2} - \boldsymbol{v_1})}{\|\boldsymbol{v_2} - \boldsymbol{v_1}\|} \times \boldsymbol{n^t}] \tag{4}$$

where $\boldsymbol{v_1}$, $\boldsymbol{v_2}$ is the first and second vertex location repectively, $\boldsymbol{n^t}$ is the normal vector calulated by:

$$\boldsymbol{n^t} = \frac{(\boldsymbol{v_2} - \boldsymbol{v_1}) \times (\boldsymbol{v_3} - \boldsymbol{v_1})}{\|(\boldsymbol{v_2} - \boldsymbol{v_1}) \times (\boldsymbol{v_3} - \boldsymbol{v_1})\|}. \tag{5}$$

We then optimize the Gaussians' local position $\boldsymbol{\mu^l}$ and local rotation $\boldsymbol{R^l}$ in triangle space instead of the global position and rotation in the original 3DGS.

Then the global rotation, scale and location of 3DGS are as follows:

$$\boldsymbol{R} = \boldsymbol{R^t}\boldsymbol{R^l}, \boldsymbol{s} = \boldsymbol{s}^l, \quad \boldsymbol{\mu} = \boldsymbol{R^t}\boldsymbol{\mu^l} + \boldsymbol{\mu^t} \tag{6}$$

where, $\boldsymbol{\mu^t}$ is the global coordinate of each triangle center. In practice, we initialize N local Gaussian points and bind them for each Gaussian point, whose initialized position is on the triangle uniformly.

This binding strategy allows 3DGS to move outside the triangle while preserving the relative position and rotation of the Gaussians after mesh manipulation.

However, following mesh manipulation, not only does the triangle center change but also the triangle shape. With the altered triangle shape, the local Gaussian position and scaling should adjust accordingly. When the triangle enlarges, it is intuitive that the local scaling and position should expand as well:

$$\boldsymbol{R} = \boldsymbol{R^t}\boldsymbol{R^l}, \boldsymbol{s} = \beta\boldsymbol{e}\boldsymbol{s}^l, \quad \boldsymbol{\mu} = \boldsymbol{e}\boldsymbol{R^t}\boldsymbol{\mu^l} + \boldsymbol{\mu^t}, \tag{7}$$

where β is a hyper-parameter, *adaption vector* $\boldsymbol{e} = [e_1, e_2, e_3]$ is designed to make sure that the global scaling $\boldsymbol{s}$ is proportionable to the triangle shape. The first axis is along the first edge, so e_1 is designed as the length l_1 of the first edge of the triangle. The second axis is along the normal direction, we set $e_2 = (0.5 * (e_1 + e_3))$. The third axis is perpendicular to the first edge, we set e_3 as the average length of the second and third edges $(0.5 * (l_2 + l_3))$.

3.4. Manipulate Gaussian Splatting through Mesh

Utilizing our triangle shape aware gaussian binding and adapting strategy, upon the completion of model training and mesh manipulation, the 3DGS is instantly manipulated and adapted. During mesh manipulation, the attributes in the local triangle space remain unchanged. The triangle rotation, position, and edge length can be calculated instantly. Therefore, the global Gaussian position, scaling, and rotation can be self-adaptively adjusted following our proposed formula. In this paper, we exhibit the 3DGS manipulation rendering outcomes, such as large-scale deformation, local manipulation, and soft-body simulation, which are driven by the manipulated mesh. In our experiments, we employ Blender to manipulate the mesh.

4. Experiments

4.1. Training Details

The first stage of our methods includes a mesh extracting stage, during which we extract triangular mesh from

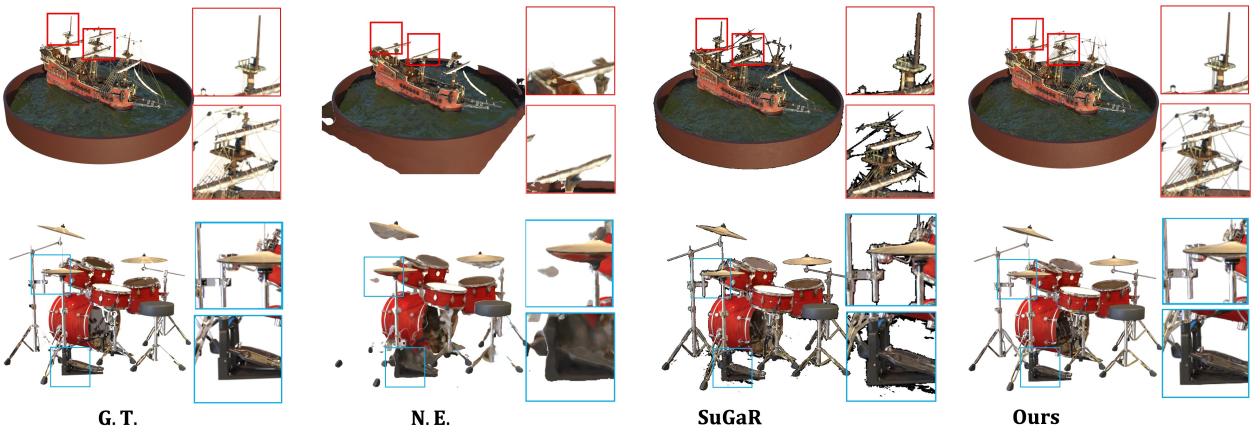

Figure 3. Visual comparison between ours, NeRF-Editing [24](N.E.) and SuGaR [10] for static rendering. It illustrates our proposed method can contain a much more accurate boundary in "*Ship*", and distinct results in "*Drums*".

Table 1. Quantitative comparison of our methods with NeRF-Editing ("*N.E.*") [50] and "*SuGaR*" [10] on NeRF Synthetic dataset in terms of SSIM, PSNR, LPIPS. "*Ours*" is with *NeuS* mesh and "*Ours(S*)*" is with the same mesh as "*SuGaR*". The bests are marked in bold, second bests are underlined. (↑ means higher is better, ↓ means lower is better)

Subject	PSNR↑				SSIM↑				LPIPS↓			
	N.E.	*SuGaR*	*Ours(S*)*	*Ours*	*N.E.*	*SuGaR*	*Ours(S*)*	*Ours*	*N.E.*	*SuGaR*	*Ours(S*)*	*Ours*
Chair	28.15	34.23	**35.68**	35.38	0.943	0.984	**0.987**	0.986	0.061	0.013	**0.010**	0.011
Drums	~~21.14~~	25.78	26.15	**26.19**	~~0.884~~	0.948	**0.953**	**0.953**	~~0.120~~	0.044	**0.038**	0.039
Ficus	~~23.82~~	32.09	35.29	**35.40**	~~0.909~~	0.973	**0.986**	**0.986**	~~0.101~~	0.025	**0.013**	**0.013**
Hotdog	32.67	36.53	**37.67**	37.49	0.969	0.983	**0.986**	0.984	0.048	0.022	**0.017**	0.019
Lego	29.16	35.05	**36.41**	36.33	0.944	0.979	**0.983**	0.982	0.074	0.018	**0.014**	0.015
Material	29.48	28.65	29.51	**29.91**	0.944	0.941	0.951	**0.956**	0.063	0.055	**0.042**	0.046
Mic	29.60	35.39	36.99	**37.46**	0.952	0.990	**0.992**	**0.992**	0.046	0.011	0.008	**0.007**
Ship	25.01	28.55	30.48	**31.01**	0.083	0.869	0.889	**0.890**	0.194	0.124	**0.097**	**0.097**
Average	-	32.06	33.52	**33.65**	-	0.958	**0.966**	**0.966**	-	0.039	**0.030**	0.031

NeuS [36] or 3DGS (Screened Poisson or Marching Cube). However, the extracted mesh always contains enormous triangles, which we try to decimate to around 300K.

With the extracted mesh, we conduct the *triangle shape aware Gaussian binding and adapting* strategy on the mesh. For each triangle, we bind $N = 3$ Gaussian on the surface initially. The Gaussian attributes are optimized subsequently with the supervision of multi-view rendering loss in the second stage. We train our model for 30K iterations in the initial stage to extract mesh and 20K iterations in the second stage. All experiments are conducted on a single NVIDIA A100 GPU.

4.2. Datasets, Metrics and Methods for Comparison

To evaluate our methods, we compare Mani-GS with previous editable novel view synthesis methods, NeRF-based editing method NeRF-Editing [50], NeuMesh [44] and 3DGS-based editing method SuGaR [10]. For the evaluation, we employ the commonly used metrics:*PSNR*, *SSIM*, *LPIPS*. We evaluate our methods mainly on the NeRF Synthetic dataset [26] and DTU dataset [16].

4.3. Evaluation

Static Rendering Table 1 provides a quantitative comparison of all NeRF Synthetic 8 cases between our method and competing methods. We conducted experiments using their official code repository. We remove two outliers ("*Drums, Ficus*") from NeRF-Editing using a strikeout. As observed, our approach outperforms all baseline methods in terms of PSNR, SSIM, and LPIPS, indicating that we achieve the best rendering quality. Our method achieves SOTA results across all 8 cases, surpassing SuGaR by at least 1.5 points in PSNR when using NeuS mesh. For a fair comparison with SuGaR, we also evaluate our binding method using the same mesh as SuGaR (denoted as "*Ours(S*)*"). Our method maintains its superior performance even with SuGaR mesh, showing comparable results to those with *NeuS* mesh.

In Figure 3, we present qualitative results of our approach and other methods in overview and zoom-in details. For SuGaR, it attempts to bind 3D Gaussians on the triangle and enforce that the attached 3DGS is as flat as possible and is closely aligned with the triangle. This binding strategy heavily depends on the accuracy of the mesh. As can be

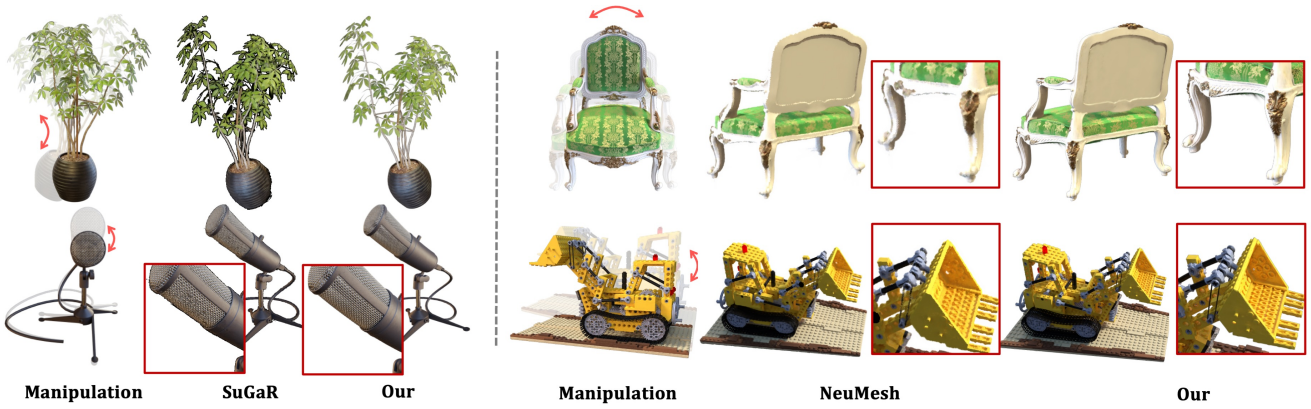

Figure 4. We offer an editing comparison between our method, SuGaR, and NeuMesh. Our approach demonstrates fewer artifacts and less blurring effects than SuGaR, and presents more abundant and distinct details compared to NeuMesh. For further details, please zoom in.

observed in the third column of Figure 3, wrong geometry leads to an inaccurate rendering, especially in the boundary region. *N.E.*[24] results are blurred and lack intrinsic details. In contrast, our method achieves higher fidelity rendering with more precise boundaries and richer details.

Methods	DTU		
	PSNR ↑	SSIM ↑	LPIPS ↓
NeuMesh [44]	28.29	0.921	0.117
SuGaR [10]	30.06	0.921	0.136
Ours	**31.50**	**0.943**	**0.088**

Table 2. We compare quantitative rendering quality with SuGaR and NeuMesh on the DTU dataset. Note that ours and SuGaR employ the same underlying mesh extracted from *poisson recon.*

We also evaluate our methods on the DTU dataset [16] with real objects, as shown in Table 2. It is important to note that we use the same base mesh as SuGaR in this dataset. As listed, we outperform both SuGaR and NeuMesh across all metrics, further demonstrating the efficacy of our 3DGS binding strategy.

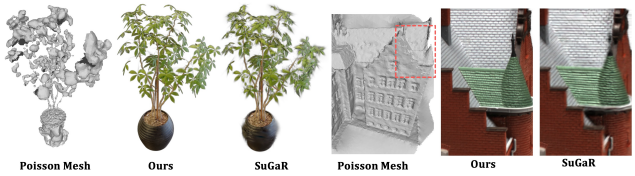

Figure 5. Visual comparison of inaccurate mesh binding and rendering. Note that ours and SuGaR employ the same underlying mesh extracted from *poisson recon.*

Binding 3DGS on inaccurate mesh. When the mesh accuracy is low, as shown on the left side of Fig 5, SuGaR produces distorted results due to the missing geometry. Conversely, when the mesh accuracy is high but still deviates from the ground truth mesh, SuGaR tends to generate blurred results because of view inconsistency with the given mesh. Thanks to our binding strategy in the triangle's local space, which allows 3DGS to move freely around the triangle, we can achieve high-fidelity rendering in both scenarios. This demonstrates that our method exhibits a high tolerance for mesh accuracy.

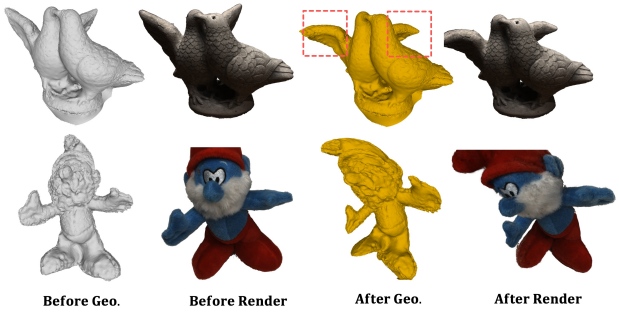

Figure 6. The manipulated geometry and corresponding rendering results of our methods on DTU dataset.

Manipulation Rendering In Figure 4, we showcase our manipulation results. In these four cases, we manipulate the underlying mesh with large deformation, the *Chair* is *stretched*, *Lego* is *tapered*, *Ficus* and *Mic* is *bent* respectively. As demonstrated in *Ficus* and *Mic*, we have more accurate boundaries and shapes, as well as distinct details. However, SuGaR can not adapt to compensate for geometry errors, which results in missing or dilated boundary rendering with distortions. For *Lego* and *Chair*, we can maintain the high rendering quality even after the large deformation, while NeuMesh showcases blurred and noised results.

In addition to the large deformation, our method also produces promising results for local manipulation and physics simulation. Here we show an example of soft body simulation. In Figure 7 row 1, we try to *blend* the red sauce and yellow sauce of *Hotdog* as shown in the blue box, which shows satisfying editing and reasonable rendering quality.

In *Drums*, we *repose* a cymbal and *elastically deform* a cymbal as shown in the blue box. After reposing and elastic deformation, we still preserve the photo-realistic rendering results. Note that the manipulation is achieved by manipulating the triangular mesh directly, the 3DGS rendering is achieved simultaneously with self-adaption.

Figure 7. Visual results of *local manipulation* (row 1) and *soft body simulation* at different timesteps.

In row 2 of Figure 7, we present the rendering results of soft body simulation at different timesteps. As observed, we can achieve soft body simulation by just transferring the mesh simulation to 3DGS, which eliminates the need for a soft body simulation algorithm dedicated to 3DGS.

4.4. Ablation Study

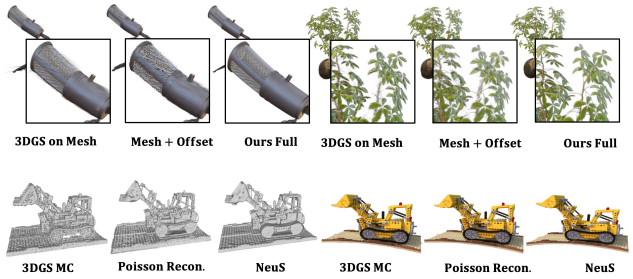

Figure 8. Ablation of binding strategy and geometry. After deformation, **(3DGS on Mesh)** shows a burring boundary, **(Mesh + Offset)** leads to significant noise and distortion, **(Ours Full)** can maintain the high fidelity rendering. In the second row, we demonstrate that even with a low-quality mesh, we can still achieve high-quality editable rendering.

We conduct ablation studies to verify the effectiveness of triangle shape-aware Gaussian binding and adapting method. We first evaluate the strategy of directly binding **3DGS on mesh**, which implies that the 3D position is fixed on the triangle. The mesh is extracted from NeuS. As shown in Table 3, the performance significantly drops, with a decrease of approximately 2.6 PSNR compared to our best model. For visual ablation in Figure 8, **3DGS on Mesh** after deformation shows a boundary with many burrs.

Next, we verify the effectiveness of adding 3D offset for 3DGS on Mesh (**Mesh + Offset**). Although the offset can enhance the fitting of 3DGS to the static scene, as demonstrated in Table 3, it fails to generate satisfactory deformation rendering results because the offset only fits the static scene and remains unchanged during subsequent deformations. Consequently, it leads to significant noise and distortion after manipulation in Figure 8, whereas our full model can maintain high-quality rendering after manipulation, demonstrating the efficacy of the 3DGS binding strategy in local triangle space.

Table 3. Quantitative ablation comparison between **3DGS On Mesh**, **Mesh + Offset**, *Ours with Marching Cube Mesh or Screened Poisson Mesh* on NeRF Synthetic dataset. (↑ means higher is better, ↓ means lower is better.)

Method	PSNR↑	SSIM↑	LPIPS↓
3DGS On Mesh	30.87	0.9521	0.0447
Mesh + Offset	32.48	0.9625	0.0341
Ours + Marching Cube Mesh	32.11	0.9602	0.035
Ours + Screened Poisson Mesh	33.42	0.9638	0.0324
Ours + NeuS Mesh	**33.45**	**0.9646**	**0.0309**

Finally, we conduct experiments with different meshes extracted using different methods. As can be observed in Figure 8 row 2, 3DGS Marching Cube Mesh (**3DGS MC**) is of low quality, including a dilated boundary and very noisy surface. And the screened poisson mesh (**Poisson Recon.**) has some unconnected regions and missing parts compared with NeuS mesh. By employing our triangle shape aware Gaussian binding and adapting method, we can still achieve 3DGS manipulation and maintain high-fidelity rendering after large deformation. In Table 3, the numerical results obtained with screened poisson mesh are only slightly lower than those obtained with NeuS mesh. When the mesh is of low quality (Marching Cube mesh), the quantitative results are approximately 1.3 PSNR lower than the best, but still 1.3 PSNR higher than only binding 3DGS inside the triangle of the best Mesh.

5. Conclusion

In this paper, we introduce a triangle shape aware Gaussian binding strategy with self-adaptation, which supports various 3DGS manipulations, maintains rendering quality, and has a high tolerance for mesh accuracy. We evaluate our methods on both synthetic and real datasets and demonstrate SOTA results.

During our experiments, we noticed that some results still exhibit distortions. When the local region of the manipulated mesh contains highly non-rigid deformations, it can result in render distortions. Additionally, during our simulation demos, we found that conducting physics simulations on meshes with more than 35K triangles can take hours.

References

[1] Rameen Abdal, Wang Yifan, Zifan Shi, Yinghao Xu, Ryan Po, Zhengfei Kuang, Qifeng Chen, Dit-Yan Yeung, and Gordon Wetzstein. Gaussian shell maps for efficient 3d human generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9441–9451, 2024. 3

[2] Chong Bao, Yinda Zhang, Bangbang Yang, Tianxing Fan, Zesong Yang, Hujun Bao, Guofeng Zhang, and Zhaopeng Cui. Sine: Semantic-driven image-based nerf editing with prior-guided editing field. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 20919–20929, 2023. 2

[3] Yue Chen, Xuan Wang, Xingyu Chen, Qi Zhang, Xiaoyu Li, Yu Guo, Jue Wang, and Fei Wang. Uv volumes for real-time rendering of editable free-view human performance. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 16621–16631, 2023. 3

[4] Zhiqin Chen, Thomas Funkhouser, Peter Hedman, and Andrea Tagliasacchi. Mobilenerf: Exploiting the polygon rasterization pipeline for efficient neural field rendering on mobile architectures. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 16569–16578, 2023. 3

[5] Yutao Feng, Xiang Feng, Yintong Shang, Ying Jiang, Chang Yu, Zeshun Zong, Tianjia Shao, Hongzhi Wu, Kun Zhou, Chenfanfu Jiang, et al. Gaussian splashing: Dynamic fluid synthesis with gaussian splatting. *arXiv preprint arXiv:2401.15318*, 2024. 3

[6] Jian Gao, Chun Gu, Youtian Lin, Hao Zhu, Xun Cao, Li Zhang, and Yao Yao. Relightable 3d gaussian: Real-time point cloud relighting with brdf decomposition and ray tracing. *arXiv preprint arXiv:2311.16043*, 2023. 4

[7] Lin Gao, Jie Yang, Bo-Tao Zhang, Jia-Mu Sun, Yu-Jie Yuan, Hongbo Fu, and Yu-Kun Lai. Mesh-based gaussian splatting for real-time large-scale deformation. *arXiv preprint arXiv:2402.04796*, 2024. 3

[8] Xiangjun Gao, Jiaolong Yang, Jongyoo Kim, Sida Peng, Zicheng Liu, and Xin Tong. Mps-nerf: Generalizable 3d human rendering from multiview images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2022. 3

[9] Xiangjun Gao, Xiaoyu Li, Chaopeng Zhang, Qi Zhang, Yanpei Cao, Ying Shan, and Long Quan. Contex-human: Free-view rendering of human from a single image with texture-consistent synthesis. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10084–10094, 2024. 3

[10] Antoine Guédon and Vincent Lepetit. Sugar: Surface-aligned gaussian splatting for efficient 3d mesh reconstruction and high-quality mesh rendering. *arXiv preprint arXiv:2311.12775*, 2023. 2, 3, 5, 6, 7

[11] Antoine Guédon and Vincent Lepetit. Gaussian frosting: Editable complex radiance fields with real-time rendering. *arXiv preprint arXiv:2403.14554*, 2024. 3

[12] Xu He, Xiaoyu Li, Di Kang, Jiangnan Ye, Chaopeng Zhang, Liyang Chen, Xiangjun Gao, Han Zhang, Zhiyong Wu, and Haolin Zhuang. Magicman: Generative novel view synthesis of humans with 3d-aware diffusion and iterative refinement. *arXiv preprint arXiv:2408.14211*, 2024. 3

[13] Liangxiao Hu, Hongwen Zhang, Yuxiang Zhang, Boyao Zhou, Boning Liu, Shengping Zhang, and Liqiang Nie. Gaussianavatar: Towards realistic human avatar modeling from a single video via animatable 3d gaussians. *arXiv preprint arXiv:2312.02134*, 2023. 3

[14] Yi-Hua Huang, Yang-Tian Sun, Ziyi Yang, Xiaoyang Lyu, Yan-Pei Cao, and Xiaojuan Qi. Sc-gs: Sparse-controlled gaussian splatting for editable dynamic scenes. *arXiv preprint arXiv:2312.14937*, 2023. 3

[15] Clément Jambon, Bernhard Kerbl, Georgios Kopanas, Stavros Diolatzis, George Drettakis, and Thomas Leimkühler. Nerfshop: Interactive editing of neural radiance fields. *Proceedings of the ACM on Computer Graphics and Interactive Techniques*, 6(1), 2023. 2

[16] Rasmus Jensen, Anders Dahl, George Vogiatzis, Engin Tola, and Henrik Aanæs. Large scale multi-view stereopsis evaluation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 406–413, 2014. 6, 7

[17] Ying Jiang, Chang Yu, Tianyi Xie, Xuan Li, Yutao Feng, Huamin Wang, Minchen Li, Henry Lau, Feng Gao, Yin Yang, et al. Vr-gs: A physical dynamics-aware interactive gaussian splatting system in virtual reality. *arXiv preprint arXiv:2401.16663*, 2024. 3

[18] Michael Kazhdan and Hugues Hoppe. Screened poisson surface reconstruction. *ACM Transactions on Graphics (ToG)*, 32(3):1–13, 2013. 4

[19] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 3d gaussian splatting for real-time radiance field rendering. *ACM Transactions on Graphics*, 42 (4), 2023. 2, 3, 4

[20] Tobias Kirschstein, Simon Giebenhain, and Matthias Nießner. Diffusionavatars: Deferred diffusion for high-fidelity 3d head avatars. *arXiv preprint arXiv:2311.18635*, 2023. 3

[21] Zhihao Liang, Qi Zhang, Ying Feng, Ying Shan, and Kui Jia. Gs-ir: 3d gaussian splatting for inverse rendering. *arXiv preprint arXiv:2311.16473*, 2023. 4

[22] Youtian Lin, Zuozhuo Dai, Siyu Zhu, and Yao Yao. Gaussian-flow: 4d reconstruction with dynamic 3d gaussian particle. *arXiv preprint arXiv:2312.03431*, 2023. 3

[23] Ruiyang Liu, Jinxu Xiang, Bowen Zhao, Ran Zhang, Jingyi Yu, and Changxi Zheng. Neural impostor: Editing neural radiance fields with explicit shape manipulation. In *Computer Graphics Forum*, page e14981. Wiley Online Library, 2023. 2

[24] Steven Liu, Xiuming Zhang, Zhoutong Zhang, Richard Zhang, Jun-Yan Zhu, and Bryan Russell. Editing conditional radiance fields. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 5773–5783, 2021. 2, 6, 7

[25] Li Ma, Xiaoyu Li, Jing Liao, Xuan Wang, Qi Zhang, Jue Wang, and Pedro V Sander. Neural parameterization for dynamic human head editing. *ACM Transactions on Graphics (TOG)*, 41(6):1–15, 2022. 3

[26] Ben Mildenhall, Pratul P Srinivasan, Matthew Tancik, Jonathan T Barron, Ravi Ramamoorthi, and Ren Ng. Nerf:

Representing scenes as neural radiance fields for view synthesis. *Communications of the ACM*, 65(1):99–106, 2021. 1, 2, 6

[27] Shenhan Qian, Tobias Kirschstein, Liam Schoneveld, Davide Davoli, Simon Giebenhain, and Matthias Nießner. Gaussianavatars: Photorealistic head avatars with rigged 3d gaussians. *arXiv preprint arXiv:2312.02069*, 2023. 3

[28] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pages 8748–8763. PMLR, 2021. 2

[29] Marie-Julie Rakotosaona, Fabian Manhardt, Diego Martin Arroyo, Michael Niemeyer, Abhijit Kundu, and Federico Tombari. Nerfmeshing: Distilling neural radiance fields into geometrically-accurate 3d meshes. *arXiv preprint arXiv:2303.09431*, 2023. 3

[30] Jingxiang Sun, Xuan Wang, Yong Zhang, Xiaoyu Li, Qi Zhang, Yebin Liu, and Jue Wang. Fenerf: Face editing in neural radiance fields. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7672–7682, 2022. 3

[31] Jiaxiang Tang, Jiawei Ren, Hang Zhou, Ziwei Liu, and Gang Zeng. Dreamgaussian: Generative gaussian splatting for efficient 3d content creation. *arXiv preprint arXiv:2309.16653*, 2023. 4

[32] Jiaxiang Tang, Hang Zhou, Xiaokang Chen, Tianshu Hu, Errui Ding, Jingdong Wang, and Gang Zeng. Delicate textured mesh recovery from nerf via adaptive surface refinement. *arXiv preprint arXiv:2303.02091*, 2023. 3

[33] Joanna Waczyńska, Piotr Borycki, Sławomir Tadeja, Jacek Tabor, and Przemysław Spurek. Games: Mesh-based adapting and modification of gaussian splatting. *arXiv preprint arXiv:2402.01459*, 2024. 3

[34] Can Wang, Menglei Chai, Mingming He, Dongdong Chen, and Jing Liao. Clip-nerf: Text-and-image driven manipulation of neural radiance fields. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3835–3844, 2022. 2

[35] Can Wang, Mingming He, Menglei Chai, Dongdong Chen, and Jing Liao. Mesh-guided neural implicit field editing. *arXiv preprint arXiv:2312.02157*, 2023. 2

[36] Peng Wang, Lingjie Liu, Yuan Liu, Christian Theobalt, Taku Komura, and Wenping Wang. Neus: Learning neural implicit surfaces by volume rendering for multi-view reconstruction. *arXiv preprint arXiv:2106.10689*, 2021. 4, 6

[37] Xiangyu Wang, Jingsen Zhu, Qi Ye, Yuchi Huo, Yunlong Ran, Zhihua Zhong, and Jiming Chen. Seal-3d: Interactive pixel-level editing for neural radiance fields, 2023. 2

[38] Guanjun Wu, Taoran Yi, Jiemin Fang, Lingxi Xie, Xiaopeng Zhang, Wei Wei, Wenyu Liu, Qi Tian, and Xinggang Wang. 4d gaussian splatting for real-time dynamic scene rendering. *arXiv preprint arXiv:2310.08528*, 2023. 3

[39] Menghua Wu, Hao Zhu, Linjia Huang, Yiyu Zhuang, Yuanxun Lu, and Xun Cao. High-fidelity 3d face generation from natural language descriptions. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4521–4530, 2023. 3

[40] Fanbo Xiang, Zexiang Xu, Milos Hasan, Yannick Hold-Geoffroy, Kalyan Sunkavalli, and Hao Su. Neutex: Neural texture mapping for volumetric neural rendering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7119–7128, 2021. 2

[41] Tianyi Xie, Zeshun Zong, Yuxin Qiu, Xuan Li, Yutao Feng, Yin Yang, and Chenfanfu Jiang. Physgaussian: Physics-integrated 3d gaussians for generative dynamics. *arXiv preprint arXiv:2311.12198*, 2023. 3

[42] Qiangeng Xu, Zexiang Xu, Julien Philip, Sai Bi, Zhixin Shu, Kalyan Sunkavalli, and Ulrich Neumann. Point-nerf: Point-based neural radiance fields. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5438–5448, 2022. 2

[43] Yuelang Xu, Benwang Chen, Zhe Li, Hongwen Zhang, Lizhen Wang, Zerong Zheng, and Yebin Liu. Gaussian head avatar: Ultra high-fidelity head avatar via dynamic gaussians. *arXiv preprint arXiv:2312.03029*, 2023. 3

[44] Bangbang Yang, Chong Bao, Junyi Zeng, Hujun Bao, Yinda Zhang, Zhaopeng Cui, and Guofeng Zhang. Neumesh: Learning disentangled neural mesh-based implicit field for geometry and texture editing. In *European Conference on Computer Vision*, pages 597–614. Springer, 2022. 2, 6, 7

[45] Ziyi Yang, Xinyu Gao, Wen Zhou, Shaohui Jiao, Yuqing Zhang, and Xiaogang Jin. Deformable 3d gaussians for high-fidelity monocular dynamic scene reconstruction. *arXiv preprint arXiv:2309.13101*, 2023. 3

[46] Yao Yao, Jingyang Zhang, Jingbo Liu, Yihang Qu, Tian Fang, David McKinnon, Yanghai Tsin, and Long Quan. Neilf: Neural incident light field for physically-based material estimation. In *European Conference on Computer Vision*, pages 700–716. Springer, 2022. 3

[47] 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. *arXiv preprint arXiv:2302.14859*, 2023.

[48] Wangbo Yu, Li Yuan, Yan-Pei Cao, Xiangjun Gao, Xiaoyu Li, Wenbo Hu, Long Quan, Ying Shan, and Yonghong Tian. Hifi-123: Towards high-fidelity one image to 3d content generation. In *European Conference on Computer Vision*, pages 258–274. Springer, 2024. 3

[49] Ye Yuan, Xueting Li, Yangyi Huang, Shalini De Mello, Koki Nagano, Jan Kautz, and Umar Iqbal. Gavatar: Animatable 3d gaussian avatars with implicit mesh learning. *arXiv preprint arXiv:2312.11461*, 2023. 3

[50] Yu-Jie Yuan, Yang-Tian Sun, Yu-Kun Lai, Yuewen Ma, Rongfei Jia, and Lin Gao. Nerf-editing: geometry editing of neural radiance fields. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 18353–18364, 2022. 1, 2, 6

[51] Fangneng Zhan, Lingjie Liu, Adam Kortylewski, and Christian Theobalt. General neural gauge fields. *arXiv preprint arXiv:2305.03462*, 2023. 2

[52] Jingbo Zhang, Xiaoyu Li, Ziyu Wan, Can Wang, and Jing Liao. Fdnerf: Few-shot dynamic neural radiance fields for

face reconstruction and expression editing. In *SIGGRAPH Asia 2022 Conference Papers*, pages 1–9, 2022. 3

[53] Zerong Zheng, Han Huang, Tao Yu, Hongwen Zhang, Yandong Guo, and Yebin Liu. Structured local radiance fields for human avatar modeling. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 15893–15903, 2022. 3

[54] Kaichen Zhou, Lanqing Hong, Enze Xie, Yongxin Yang, Zhenguo Li, and Wei Zhang. Serf: Fine-grained interactive 3d segmentation and editing with radiance fields. *arXiv preprint arXiv:2312.15856*, 2023. 2

[55] Xiaoyu Zhou, Zhiwei Lin, Xiaojun Shan, Yongtao Wang, Deqing Sun, and Ming-Hsuan Yang. Drivinggaussian: Composite gaussian splatting for surrounding dynamic autonomous driving scenes. *arXiv preprint arXiv:2312.07920*, 2023. 3

[56] Yiyu Zhuang, Hao Zhu, Xusen Sun, and Xun Cao. Mofanerf: Morphable facial neural radiance field. In *European conference on computer vision*, pages 268–285. Springer, 2022. 3

[57] Yiyu Zhuang, Yuxiao He, Jiawei Zhang, Yanwen Wang, Jiahe Zhu, Yao Yao, Siyu Zhu, Xun Cao, and Hao Zhu. Towards native generative model for 3d head avatar. *arXiv preprint arXiv:2410.01226*, 2024. 3

[58] Wojciech Zielonka, Timur Bagautdinov, Shunsuke Saito, Michael Zollhöfer, Justus Thies, and Javier Romero. Drivable 3d gaussian avatars. *arXiv preprint arXiv:2311.08581*, 2023. 3

[59] Matthias Zwicker, Hanspeter Pfister, Jeroen Van Baar, and Markus Gross. Ewa splatting. *IEEE Transactions on Visualization and Computer Graphics*, 8(3):223–238, 2002. 4