High-Quality Geometry and Texture Editing of Neural Radiance Field

Soongjin Kim[1] Jooeun Son[1] Gwangjin Ju[1] Joo Ho Lee[2] Seungyong Lee[†1]

[1]POSTECH, Korea
[2]Sogang University, Korea

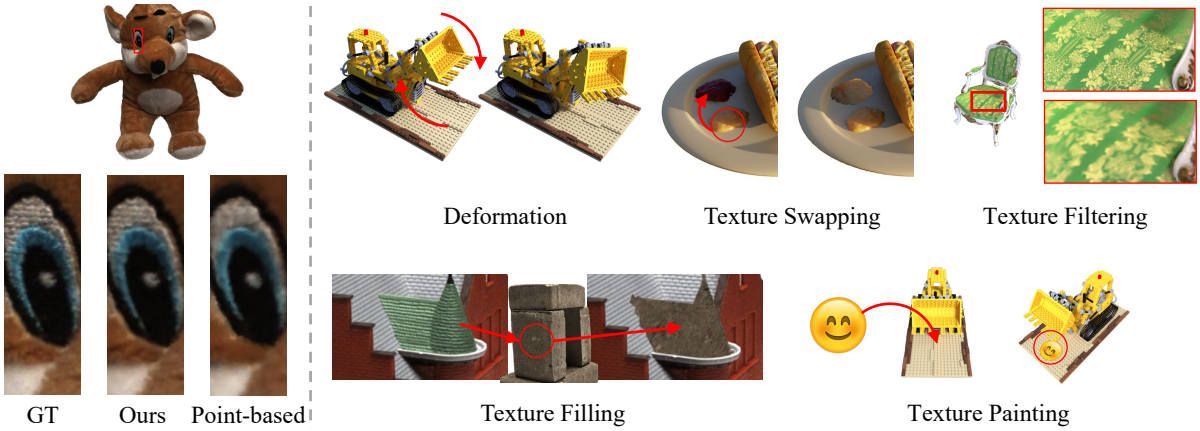

Figure 1: *We present a novel surface-aligned volumetric radiance field for high-quality reconstruction and flexible scene editing. (Left) Our surface-aligned volumetric radiance field successfully captures high-frequency details of the scene appearances compared to the point-based representation [YBZ*22]. (Right) Our approach allows for neural volumetric representation to be used for various scene editing operations including geometric deformation, texture swapping, texture filtering, texture filling, and texture painting.*

Abstract

Recent advances in Neural Radiance Field (NeRF) have demonstrated impressive rendering quality reconstructed from input images. However, the density-based radiance field representation introduces entanglement of geometry and texture, limiting the editability. To address this issue, NeuMesh proposed a mesh-based NeRF editing method supporting deformation and texture editing. Still, it fails reconstructing and rendering fine details of input images, and the dependency between rendering scheme and geometry limits editability for target scenes. In this paper, we propose an intermediate scene representation where a near-surface volume is associated with the guide mesh. Our key idea is separating a given scene into geometry, parameterized texture space, and radiance field. We define a mapping between xyz-coordinate space and uvh-coordinate system defined by combination of mesh parameterization and the height from mesh surface to efficiently encode the near-surface volume. With the surface-aligned radiance field defined in the near-surface volume, our method can generate high quality rendering results with high frequency details. Our method also supports various geometry and appearance editing operations while preserving high rendering quality. We demonstrate the performance of our method by comparing it with the state-of-the-art methods both qualitatively and quantitatively and show its applications including shape deformation, texture filling, and texture painting.

CCS Concepts
• ***Computing methodologies*** → *Image-based rendering;*

1. Introduction

Neural rendering has struck computer graphics society with its remarkable ability to capture realism and the versatility. One of the

† Corresponding author

DOI: 10.2312/pg.20241318

seminal works, Neural Radiance Field (NeRF) [MST*20], trains a single multi-layered perception (MLP) network to learn the volumetric radiance field of a given scene, achieving high fidelity of reconstruction. On the other hand, NeRF-based representations suffer from two major limitations when applied for scene editing. Entanglement of geometry and texture within the volumetric radiance field makes the NeRF representation not compatible with traditional scene editing operations such as geometry deformation and texture replacement. In addition, for scene editing, the NeRF representation requires modification of network parameters as the rendering output depends on all network parameters.

To facilitate the editing within the NeRF representation, several approaches have been proposed recently. Most methods handle geometry editing as transformation of object space [JXX*21, YZX*21], volumetric control with a deformation graph [SSP07, IZN*16, NFS15], or mesh deformation [YSL*22, XH22] while neglecting texture editing. NeuMesh [YBZ*22] proposes a mesh-based editable representation that supports both geometry and texture editing. It encodes the volumetric radiance field using a vertex-based feature representation that enables local editing of geometry and texture by moving vertices and modifying vertex features. However, as features are stored at mesh vertices, the amount of appearance details are limited by the number of vertices and textures can be edited only through interpolated vertex features, prohibiting direct manipulation of texture details.

In this paper, we propose an intermediate scene representation where a near-surface volume is associated with the guide mesh. Our key idea is separating a given scene into geometry, parameterized texture space, and radiance field. We disentangle the volumetric radiance field from the geometry by defining the field in the near-surface volume surrounding the guide mesh. The near-surface volume is parameterized by 2D texture coordinates (u, v) of guide mesh surface points and the heights h from the mesh surface. We divide the near-surface volume into a set of well-aligned tetrahedrons and use barycentric interpolation to define the mapping of a given xyz location to the corresponding uvh coordinates on the fly.

With the surface-aligned radiance field defined in the near-surface volume, our method can generate high quality rendering results with high frequency details while handling textures with higher resolutions than vertex-based representation. The geometry editing can be easily handled by manipulating the guide mesh, where the edited appearances are immediately obtained by changes of the mapping between xyz and uvh coordinates. In addition, independently of the geometry, texture editing operations can be directly applied to the radiance field in the near-surface uvh volume. We demonstrate the performance of our method by comparing it with the state-of-the-art methods both qualitatively and quantitatively and show various applications including shape deformation, texture filling, and texture painting.

Main contributions of our paper can be summarized as follows:

- We propose a novel mesh-based neural rendering method that supports effective geometry and texture editing by associating a near-surface volume with the guide mesh.
- Our method achieves better rendering quality than a vertex-feature-based NeRF editing method by encoding the surface-aligned radiance field in the near-surface volume.
- Our decoupled representation of geometry and appearance enables various applications such as shape deformation, texture filling, and texture painting.

2. Related Work

2.1. Neural Novel-view Synthesis

The big success of adopting MLP networks [MST*20] into image-based novel-view synthesis [BUE01, LH96] has attracted tremendous attention to neural rendering. The coordinate-based neural network with positional encoding achieves photo-realistic quality, albeit with long training time. To mitigate the training issue, spatial structures have been explored to encode the features of a scene, such as voxel grids [FKYT*22, SSC22], point clouds [XXP*22], and hash grids [MESK22]. These feature-based grid representations require relatively short training time while providing comparable reconstruction quality. Kerbl et al. [KKLD23] proposes an orthogonal approach, where a scene is represented as a set of 3D spherical Gaussian distributions that are splatted to create high-quality renderings at interactive frame rates.

2.2. Scene Editing with Neural Representations

NeRF-based representations [MST*20, FKYT*22, MESK22] are hard to edit due to the entanglement of geometry and texture in the volumetric radiance field and the nonlocality of network parameters. To resolve these issues, various editing approaches are proposed by increasing the locality of parameters and disentangling geometry from the volumetric radiance field.

For geometry editing, learning the neural representation for each object position [JXX*21, YZX*21] enables object translation on the fly. The deformation graph [SSP07, IZN*16, NFS15] was also used to deform the object volume by moving control points, handling non-rigid object motion [YSL*22, XH22]. NeRF-editing [XH22] increases the resolution of the deformation graph to be close to the object geometry for full control of geometry editing.

To associate the neural representation with the texture space, the volumetric radiance field should learn the proper surface geometry and its texture mapping. NeuTex [XXH*21] encourages the neural representation to construct the surface geometry and map it to a unit sphere associated with a texture space [Pau98], where the surface radiance field is encoded. This indirect texture mapping enables texture editing of objects by manipulating the texture map on the sphere while restricting the object's geometric topology.

NeuMesh [YBZ*22] supports free-form editing for both geometry and texture without constraints on scenes. It first computes a guide mesh from the first round of neural reconstruction, then learns vertex features of the guide mesh to encode the near-surface volumetric radiance field. In that approach, the resolution of mesh vertices is crucial to both the reconstruction quality [WS23] and the degree of fine editing. Furthermore, to support texture editing, the editing on the image space should be back-projected to the volume by retraining the radiance field represented by vertex features. In contrast, our method decouples the geometry and texture representations so that each of them can be independently modified for desired editing operations. Geometry editing is achieved by our

surface-aligned representation of the radiance field. On-the-fly texture editing is enabled by the uvh representation of the near-surface volume that allows editing operations in the uv domain.

DE-NeRF [WSLG23] extends NeuMesh [YBZ*22] to incorporate relighting, enabling separate editing of texture and lighting. DE-NeRF inherits the limitations of NeuMesh, particularly in terms of quality degradation. Seal-3D [WZY*23] leverages the locality of the feature lattice grid to accelerate the editing process. However, it still requires global fine-tuning to reduce visual artifacts and is limited to smooth volumetric deformations. EditNeRF [LZZ*21] utilizes shape semantics to support part-aware editing, facilitating part colorization and part deletion using the category-shape prior. However, it can only edit objects from pre-trained categories.

3. Surface-aligned Volumetric Radiance Field

Our surface-aligned volumetric radiance field has two major parts: constructing a surface-aligned volume and learning a near-surface volumetric radiance field. In the volume construction part, we first obtain a guide mesh and then construct the surface-aligned volume by placing tetrahedra near the surface. The neural representation defined in the volume learns the volumetric appearance near the surface via differentiable volumetric rendering. Our surface-aligned volumetric radiance field enables scene editing of geometry and texture, as demonstrated in Section 5.

3.1. Surface-Aligned Volumetric Space

To reconstruct the volumetric appearance of the scene, we build a surface-aligned volume V surrounding the guide mesh M. We represent volume V using a set of tetrahedrons that are constructed based on the triangles of mesh M. Figure 2 illustrates the schematic overview of our surface-aligned volume.

Guide mesh reconstruction We first estimate the implicit representation of a given scene by a geometry reconstruction method [WHH*23]. We then extract the guide mesh M using the marching cube algorithm [LC98]. As the initial M may contain small and skewed triangles hindering mesh-based editing, we refine it by mesh simplification [GH97] and remeshing [HDD*93] to enhance the regularity. Finally, we perform mesh parameterization [ZSGS04] to obtain the (u,v) space on the base mesh surface.

Surface-aligned volume representation Differently from an ordinal axis-aligned volume used for neural rendering to cover 3D space [MESK22, CXG*22], our surface-aligned volume V covers only the space surrounding the given guide mesh M (Figure 3(a)). We parameterize a near-surface point $\mathbf{x} \in V$ as a 3D vector composed of 2D surface location (u,v) and 1D offset height $\tilde{h}$ (Figure 3(b)):

$$\mathbf{x} = \mathbf{x}_0(u,v) + \tilde{h} \cdot \mathbf{d}(u,v), \tag{1}$$

where $\mathbf{x}_0$ is a mesh parameterization function that maps a point in the 2D plane to a 3D point on M and $\mathbf{d}(u,v)$ is a 3D offset direction determined by barycentric interpolation of the offset directions of triangle vertices. $\mathbf{x}_0(u,v)$ is the corresponding mesh point and $\tilde{h}$ denotes the height from the guide mesh surface. We bound the height $\tilde{h}$ for each vertex, expanding a triangle to a 3D triangular frustum, where offset directions of vertices determine the shape of the frustum. We choose vertex normal as the offset direction at each vertex, like [CVM*96], to avoid overlapping between frustums of adjacent mesh triangles. As a result, the surface-aligned volume V consists of connected frustums containing the triangles of M.

Setting per-vertex height range In our construction of V, a constant height offset might twist frustums as shown in Figure 4 (a). To circumvent frustum flipping, we set a per-vertex height range. For a vertex v, we move v together with neighbor vertices using the same height along the offset directions until the minimum distance of the moved neighbor vertices from the moved v becomes below the threshold. We then set the height as the upper limit of height range of vertex v. In our frustum construction, vertices can move in both positive and negative offset directions, and we set the height ranges for both directions.

Frustum subdivision into tetrahedrons We represent our surface-aligned volume V using a set of tetrahedra, where the tetrahedral representation provides a natural way of specifying points in V using barycentric coordinates. However, the side faces of a frustum constructed from a triangle in M could be non-planar due to different offset directions of vertices. If we simply approximate each frustum independently with tetrahedra constructed using frustum corners, the resulting set of tetrahedra may miss or include twice portions of some frustums, as non-planar side faces are shared among two frustums constructed from adjacent triangles in M. To remove such cases, we decompose each frustum into three tetrahedra in a systematic order.

For an edge (v_1, v_2) of M, let the moved edge with the largest vertex offsets be (v_1^t, v_2^t) and the smallest offsets be (v_1^b, v_2^b) (Figure 4 (b)). There are two possibilities to connect the moved edges with two triangles, i.e., triangles $(v_1^t, v_2^t, v_1^b), (v_2^t, v_1^b, v_2^b)$ or $(v_1^t, v_1^b, v_2^b), (v_1^t, v_2^t, v_2^b)$ (Figure 4 (c)). When the division choices for this case differ for two frustums constructed from the adjacent triangles sharing the edge (v_1, v_2) in M, the tetrahedral volume defined by $(v_1^t, v_2^t, v_2^b, v_1^b)$ becomes missed or covered twice in the resulting set of tetrahedra (Figure 4 (d)). To handle this issue, we utilize the vertex ids of mesh M. That is, the same division choice is made for an edge (v_1, v_2) regardless of the triangles containing the edge, e.g., connecting v_1^t to v_2^t if the vertex id of v_1 is smaller than that of v_2. With this consistent subdivision, volume V can be decomposed into a set of tetrahedra without holes and overlaps.

3.2. Volumetric Appearance in Surface-aligned Volume

Converting into the uvh-space We first set uvh-coordinates of vertices in frustums F. To this end, vertices of frustums inherit their uv-coordinates from their corresponding vertices on the guide mesh. For height h, vertices on the bottom layer have value 0 while vertices on the top layer have value one.

The position of a point x inside a frustum F can be converted into uvh-coordinates. Considering that F has been subdivided into three tetrahedrons, we first check which tetrahedron contains x and calculate the barycentric coordinates of x for the tetrahedron. Then, as we know the uvh-coordinates of the vertices of F, we can compute

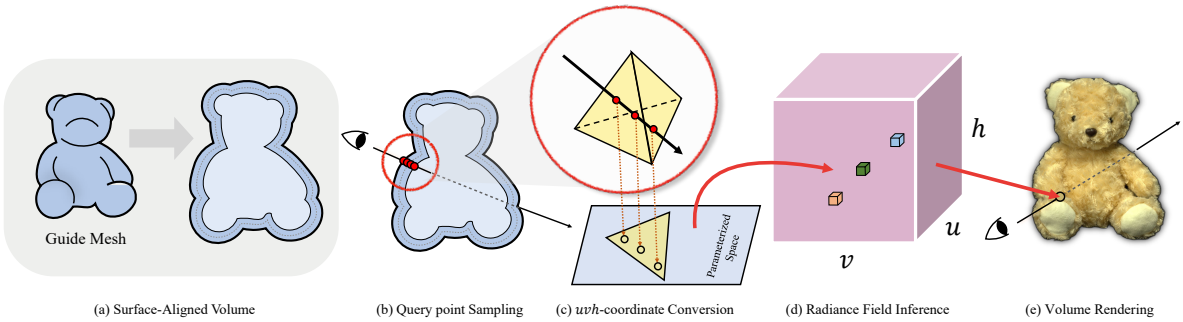

Figure 2: *Overview. Our method inputs a guide mesh geometry and registered multi-view images. (a) The surface-aligned volumetric space is constructed from the guide mesh, by offsetting in the vertex offset directions. (b) In our method, query points are sampled only inside the surface-aligned volume using a ray marching method. (c) Each query point inside the volume is mapped into a tetrahedron and converted into uvh-coordinates by referring to the mesh parameterized space. (d) The query points are fed into the radiance field to infer RGB and density values. (e) The pixel color of a view ray is determined by the volume rendering equation applied to query points.*

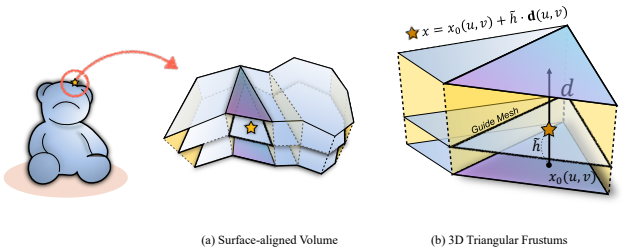

Figure 3: *Surface-aligned coordinate system. A near-surface point (star) is parameterized by its 2D surface location (u,v) and height offset h. (a) The surface-aliened volume covers near-surface points within the offset range. (b) Triangular frustums covering the surface-aliened volume.*

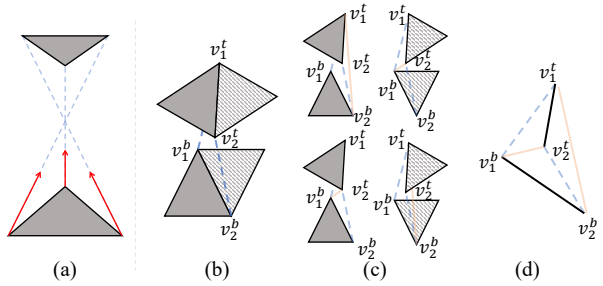

Figure 4: *Possible robustness problems on construction a set of tetrahedrons. (a) Frustum flipping might appear when offset directions are not parallel. Connecting top and bottom triangles in 3D (b) has several choices as illustrated in (c). If the division choices differ, the volumes in (d) becomes missed or covered twice.*

the uvh-coordinates $\mathbf{x}^u$ of x by barycentric interpolation of the uvh-coordinates of tetrahedron vertices;

$$\mathbf{x}^u = \sum_{i=1}^{i=4} w(i) * \mathbf{u}[i], \tag{2}$$

where $w(\cdot)$ is a barycentric weight and $\mathbf{u}[\cdot]$ is the uvh-coordinates of a tetrahedron vertex that have been obtained in the preprocessing step of guide mesh construction.

Volumetric rendering To estimate color $\hat{C}(\mathbf{p})$ at pixel $\mathbf{p}$, a sequence of 3D query points $\mathbf{x}_i$ for $i \in [1,\dots,N]$ are sampled along the view ray corresponding to $\mathbf{p}$. Then, we check whether each query point $\mathbf{x}_i$ is located inside or outside the surface-aligned volume V. If inside, the uvh-coordinates of $\mathbf{x}_i$ are determined to obtain the color $\mathbf{c}(\mathbf{x}_i,\mathbf{e})$ and density $\sigma(\mathbf{x}_i)$ in the uvh-space, where $\mathbf{e}$ is the view direction. In our implementation, we use the TensoRF representation [CXG*22] to encode the color and density in the uvh-space. Finally, we compute the volumetric rendering equation [KVH84] for the view ray at $\mathbf{p}$:

$$\hat{C}(\mathbf{p}) = \sum_{i=1}^{N} T_i (1 - \exp(-\sigma(\mathbf{x}_i)\delta_i) 1_{x_i \in V}) \mathbf{c}(\mathbf{x}_i, \mathbf{e}), \tag{3}$$

where $\mathbf{x}_i$ is the i-th query point, δ_i is the depth difference between the i-th and $i+1$-th query points, and T_i is the transmittance between the camera and the i-th query point. $1_{x_i \in V}$ is 1 when x_i is inside of V and 0 otherwise. δ_i uses the distance in the xyz-space for robustness. For $\mathbf{e}$, we use the original view direction in the xyz-space. Please refer to [MST*20] for more details.

Near-surface ray marching Since we already have prior knowledge of geometry, efficient sampling strategy can be utilized. We choose ray traversal sampler of [LGTK23], and the sampler finds valid voxel locations inside the bounding box and marches the ray with a fixed step size while skipping empty voxels. In our case, a masking voxel grid cell is denoted as valid when contained in the volume V. For each grid cell, the ids of the tetrahedra overlapping with the grid cell are stored so that a query point in the cell can be converted to the barycentric coordinates of the tetrahedron containing the point. Sometimes more than one tetrahedron might contain the same query point in a concave region or after deformation. When such collision happens, the higher priority is given to the tetrahedron where the point has the smaller absolute height value, reflecting the preference on closer points near the surface.

4. Surface-aligned Scene Editing

4.1. Geometry Editing

Since the appearance of the scene has been captured in the surface-aligned volume V, we can perform geometry editing by directly deforming the guide mesh M. Any kind of deformation can be combined with our method, such as as-rigid-as-possible (ARAP) deformation [SA07] and cage deformation [JMD*07] (Figure 5). At run-time, the vertex normals of the deformed M are recalculated by averaging face normals. The surface-aligned volume V' is then recalculated using the updated offset directions and the set of tetrahedrons are updated accordingly. With the updated coordinate conversion setup, 3D query points $\mathbf{x}_i$ are converted into the corresponding uvh-coordinates. The color and density values are estimated from the radiance field pre-trained in V. Finally, volume rendering using Eq. (6) generates the updated appearance of the scene, achieving high fidelity geometry editing.

4.2. Texture Editing

Our method supports various texture editing operations by directly working in the uvh-space of the surface-aligned volume V used for encoding the near-surface radiance field.

Texture filling To replace some region of a source scene by a texture patch of the target scene, we obtain the uv-coordinates of the target texture patch and save it in a mapping image. When rendering the result, we first sample and convert query points with original uv-coordinates in the source scene. The color and the density outside the modified region are estimated with the source radiance field. For the query points inside the modified area, we refer the pre-calculated mapping image for converting uv-coordinates from source uv to target uv. The height values are not changed during the conversion. When creating the mapping image, we use a repetitive texture pattern to fill the whole image. Finally, our updated rendering equation for pixel $\mathbf{p}$ is as follows:

$$\hat{C}(\mathbf{p}) = \sum_{i=1}^{N} T_i (1 - \exp(-\sigma_{\mathbf{s}}(\mathbf{x}_i)\delta_i) 1_{x_i \in V}) \mathbf{M}(\mathbf{x}_i, \mathbf{e}), \quad (4)$$

$$\mathbf{M}(\mathbf{x}_i, \mathbf{e}) = \begin{cases} \mathbf{c_t}(U(\mathbf{x}_i, \mathbf{N}), \mathbf{e}) & \text{if } \mathbf{x}_i \in V_{\mathbf{s}} \\ \mathbf{c_s}(\mathbf{x}_i, \mathbf{e}) & \text{otherwise} \end{cases} \quad (5)$$

where $\sigma_{\mathbf{s}}$ is the densities of source scene, and $\mathbf{c_s}$ and $\mathbf{c_t}$ are colors of source and target scenes, respectively. $V_{\mathbf{s}}$ is the user-specified modified region in the source space. $U(\cdot, \mathbf{N})$ is the uv conversion operation using a mapping image $\mathbf{N}$.

Texture swapping In texture swapping, we replace some region of a scene by other patch of the same scene. Similarly to texture filling, we specify uv-coordinates for the modified region and create a mapping image $\mathbf{N}$. The main difference is that color and density estimations will be performed in the same radiance field. We modify Eq. (4) as follows;

$$\hat{C}(\mathbf{p}) = \sum_{i=1}^{N} T_i (1 - \exp(-\sigma_{\mathbf{s}}(U(\mathbf{x}_i, \mathbf{N}))\delta_i) 1_{x_i \in V}) \mathbf{c_s}(U(\mathbf{x}_i, \mathbf{N}), \mathbf{e}). \quad (6)$$

Figure 5: *Geometric deformation. Our mesh-based representation can be associated with any mesh-based deformation methods, such as As-Rigid-As-Possible deformation (top) and cage deformation (bottom). Original and deformed cage wireframes are visualized.*

Texture painting Unlike NeuMesh [YBZ*22] requiring additional optimization, we can paint on a scene without further training by following the texture modification operation explained above. We can consider painting as a special case of texture swapping, where a mapping image N contains RGB color values instead of uv-coordinate mapping. To improve the quality of painting, we apply weighted blending using the alpha channel of image N. The mapping function for texture paining is defined as follows;

$$\mathbf{M}(\mathbf{x}_i, \mathbf{e}) = \mathbf{c_s}(\mathbf{x}_i, \mathbf{e})(1 - U(\mathbf{x}_i, \mathbf{N})_\alpha) + U(\mathbf{x}_i, \mathbf{N})_{rgb} * U(\mathbf{x}_i, \mathbf{N})_\alpha \quad (7)$$

where subscripts α and rgb represent the alpha and RGB channels of image $\mathbf{N}$, respectively.

Texture filtering As our method represents the scene appearance using the surface-aligned volume V, the scene's texture can be directly manipulated by performing filtering operations in the volume V. For example, smoothing and downsampling operations, often used for anti-aliasing or level-of-detail (LoD) representation, can be readily applied to the sampled points in V.

5. Results

We utilize TensoRF [CXG*22] as our main radiance field representation. In Section 5.3, we show the additional result with another baseline. When we train radiance fields, the loss function is the same as the baseline model. The parameter size and training iterations are also the same as the baseline model.

Dataset and experiment setup We validate our method on two public datasets, DTU MVS [JDV*14] and NeRF synthetic [MST*20]. For DTU dataset, we experimented with 7 scenes, each with images of 1600 × 1200 resolution. We collect every 8th frame as the test set. Other images are used as the training set. For NeRF-synthetic, we make comparisons on the same four scenes as in NeuMesh [YBZ*22], using the official test/training sets.

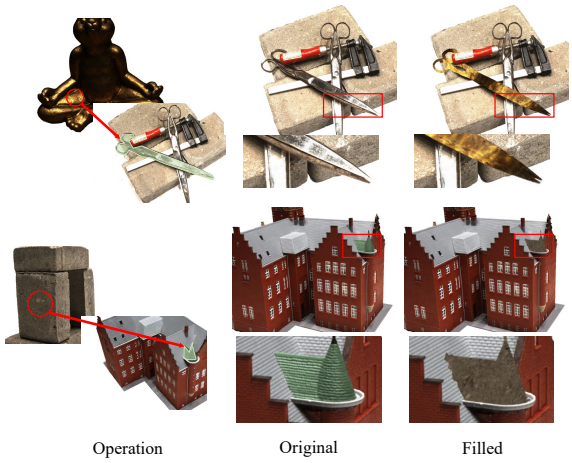

Figure 6: *Texture filling. The texture in a source scene region (green) is completed using the texture from a target scene region (red circle). Our texture filling successfully transfers the texture from the target to the source region.*

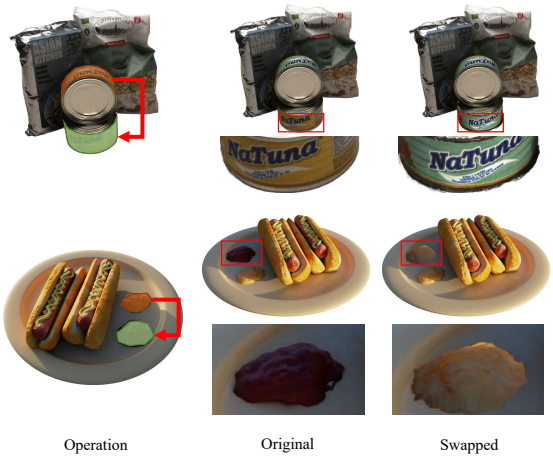

Figure 7: *Texture swapping. The texture in the green target region is replaced by the texture in the orange source region.*

5.1. Scene Editing

Geometric deformation Figure 5 shows the result of geometry deformation using our representation. Our method changes the scene geometry by deforming the guide mesh. The appearance in the reference volume is completely transferred to the deformed surface-aligned volume by redefining the tetrahedral volume for each triangle of the guide mesh. Our mesh-based representation is compatible with various mesh deformation methods.

Texture filling Figure 6 illustrates results of texture filling. Instead of directly copying and pasting the texture information, we add a mapping layer that redirects the filling region in the source scene to the *uv* space in the target scene.

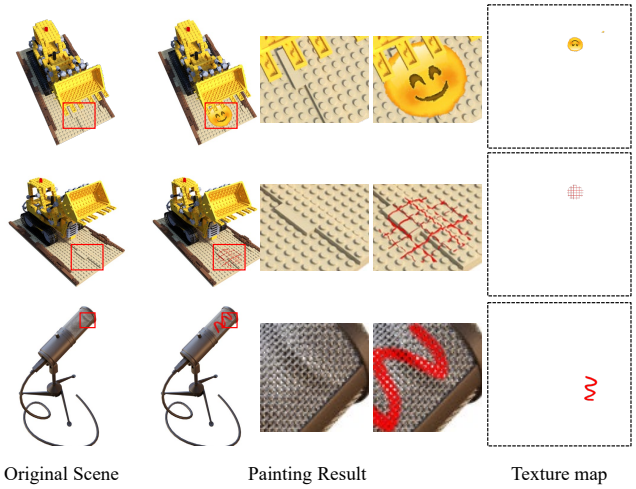

Figure 8: *Texture painting. We attach a smile emoji (top) and red grid pattern (middle) onto the lego plate. For the mic scene (bottom), we draw a red scribble on the head part.*

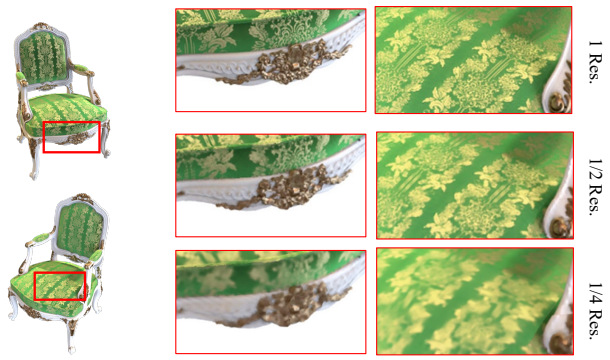

Figure 9: *Texture filtering. We downsample the tensorial representation into half and quarter resolutions. As the resolution becomes smaller, high-frequency information is progressively filtered out.*

Texture swapping Figure 7 illustrates results of our texture swapping. By redirecting *uv*-coordinates of a source region to the target region in the same scene, the source appearance is swapped with the target appearance instantly. Our texture swapping successfully modifies the appearance while keeping the structure of the scene.

Texture painting We demonstrate our painting operation in Figure 8. To this end, we add one additional RGB texture map associated with the guide mesh. The RGB texture map can be edited in the texture space by finding the corresponding *uv* point for each image pixel. Texture-space painting allows painting in an occluded or narrow scene region that the camera cannot easily see. Our texture-only painting preserves the volumetric appearance of the scene, e.g., holes on the mic in Figure 8. In addition, our texture painting

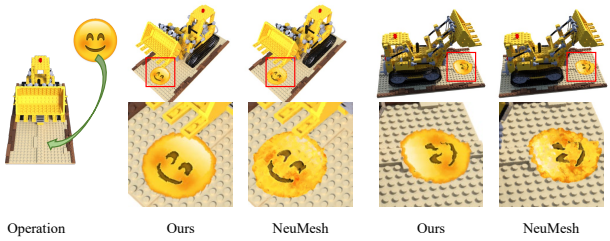

Figure 10: *Ours vs. NeuMesh [YBZ*22] in texture painting.*

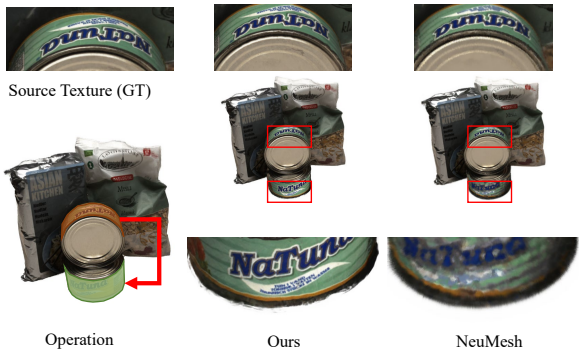

Figure 11: *Ours vs. NeuMesh [YBZ*22] in texture swapping.*

can be done on the fly even with a varying texture. In contrast, NeuMesh [YBZ*22] re-trains the related features and the rendering network with the painting condition. It requires sufficient training iterations to achieve the previous reconstruction quality after painting operation.

Texture filtering Figure 9 shows texture filtering results with the TensoRF representation used for the surface-aligned volume. We downsample the *uv*-plane tensor and perform bilinear-interpolation to compute the scene appearance.

5.2. Comparison with Previous Methods

Comparison against NeuMesh Our method holds significant advantages in texture editing compared to NeuMesh [YBZ*22]. NeuMesh does not require mesh parameterization because it encodes the near-surface volumetric radiance field into vertex features. However, due to the absence of mesh parameterization, all surface texture editing must be projected onto the features of relevant vertices by retraining them. This process may incur unintended appearance artifacts on nearby non-edited regions during texture editing. In contrast, our texture editing is directly performed in the surface-aligned volume without any additional operation, such as projection onto vertices, and prevents any quality degradation on the appearances of non-edited regions. Besides, our method supports direct texture editing on the *uv*-space, as shown in Figure 10, allowing editing operation on an occluded or narrow scene region.

Figure 11 demonstrates the performance of our method against NeuMesh in texture swapping. NeuMesh requires an additional learning process to swap the texture in the filled region, resulting in

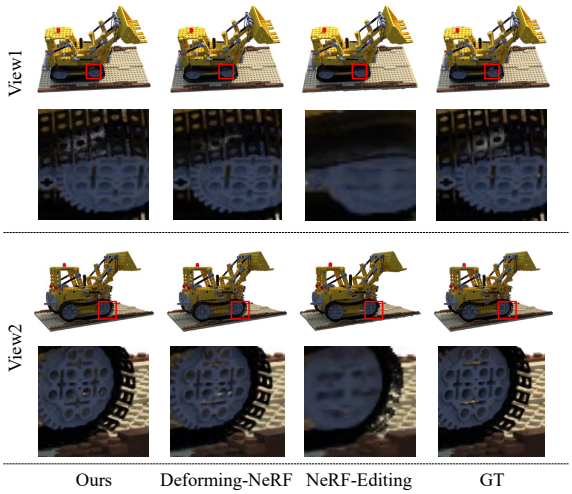

Figure 12: *Qualitative comparison against other deformation methods on NeRF Synthetic dataset before deformation. Our method reconstructs the volumetric radiance field with high fidelity.*

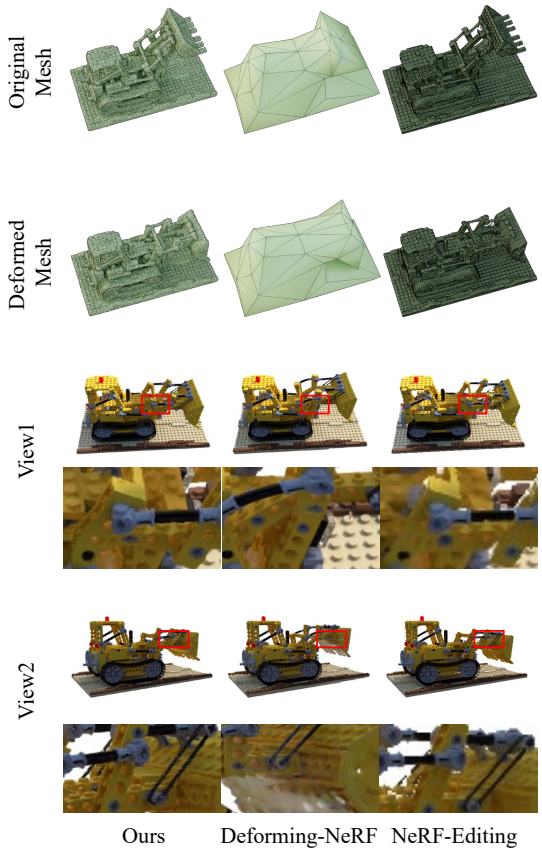

Figure 13: *Comparison against other deformation methods in terms of geometric deformation. Our representation deforms the scene with respect to the scene structure while achieving the high rendering quality.*

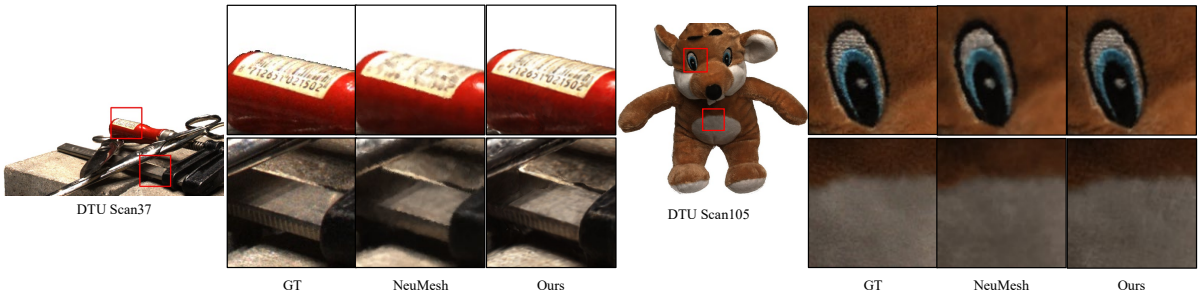

Figure 14: *Comparison on two scenes from the DTU dataset. We compare our method with NeuMesh [YBZ*22]. Our method can produce better view synthesis results, especially on texture regions.*

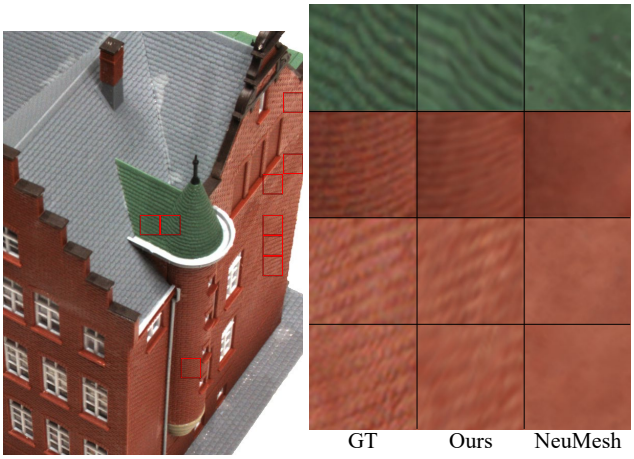

Figure 15: *Comparison on high-gradient image patches. (a) Patches with highest numbers of edge pixels. Top 1% are automatically selected via the Canny edge detector, marked as red boxes. (b) Visual comparison using top four patches.*

the degenerated quality of the transferred texture. In contrast, our method keeps the previous reconstruction quality and transferred the texture on the fly.

Comparison with deformation methods We compare the performance of our method with the public versions of other neural deformation methods, Deforming-NeRF [XH22] and NeRF-Editing [YSL*22], using the *lego* dataset (Figure 12). The reconstruction quality depends on which representation is used for scene appearance. Our method using TensoRF representation [CXG*22] and Deforming-NeRF with plenoxel [FKYT*22] representation use the density-based radiance field which captures scene details properly. NeRF-editing associated with the NeuS [WLL*21] representation uses SDF fields for representing scenes and often fails to reconstruct fine appearance details.

Figure 13 illustrates the degree of flexibility for neural deformation methods including ours. As Deforming-NeRF uses a few control points to warp the volume, it bends or stretches the scene structure without preserving the scene rigidity. Our method achieves structure-aware deformation as flexible as NeRF-editing while achieving better rendering quality.

Table 1: *Comparison of the rendered image quality on DTU dataset. The best values for each data scope are marked as bold. We present the average value for all test scenes.*

Data Scope	Method	PSNR ↑	LPIPS ↓	SSIM ↑
All	NeRF	25.76	0.2946	0.9236
	NeuMesh	26.90	0.2065	0.9554
	Ours	**27.80**	**0.1354**	**0.9707**
Top 1%	NeRF	23.87	0.4503	0.4929
	NeuMesh	24.08	0.4134	0.5056
	Ours	**27.08**	**0.1948**	**0.7769**
Top 5%	NeRF	23.83	0.4445	0.5118
	NeuMesh	24.30	0.4003	0.5332
	Ours	**27.33**	**0.1916**	**0.7877**
Top 10%	NeRF	23.96	0.4372	0.5228
	NeuMesh	24.46	0.3915	0.5534
	Ours	**27.51**	**0.1888**	**0.7956**

Table 2: *Comparison of the rendered image quality on NeRF synthetic dataset. The best values are marked as bold. We present the average value for all test scenes. The method marked with * means that image metric values are from the original paper.*

Method	PSNR ↑	LPIPS ↓	SSIM ↑
NeRF*	33.66	0.061	0.963
NeuMesh*	30.945	0.043	0.951
DE-NeRF*	29.18	0.035	0.959
Ours	**36.05**	**0.028**	**0.983**

Comparison of rendered image quality We compare the rendered image quality of our method with the state-of-the-art radiance field editing methods (NeuMesh [YBZ*22] and DE-NeRF [WSLG23]) and novel-view synthesis method (NeRF [MST*20]). Our method outperforms all of the methods across the datasets, as shown in Tables 1 and 2. Figure 14 shows qualitative results where our method captures high-frequency details.

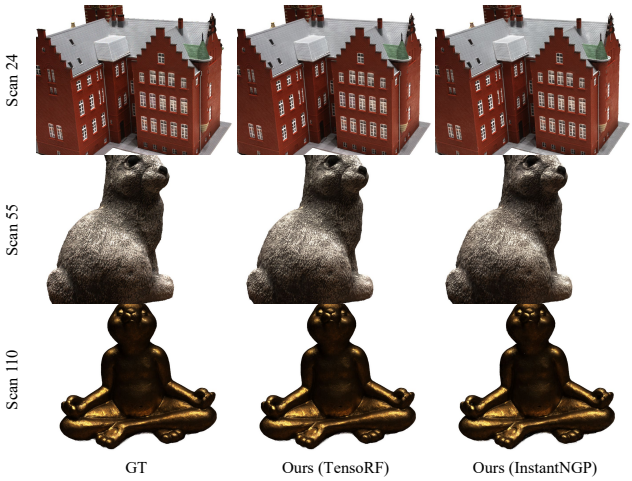

Figure 16: *Appearance model compatibility. Our approach can be combined with any volumetric scene representation, including the tensorial representation [CXG*22] and the multi-resolution hash grid representation [MESK22].*

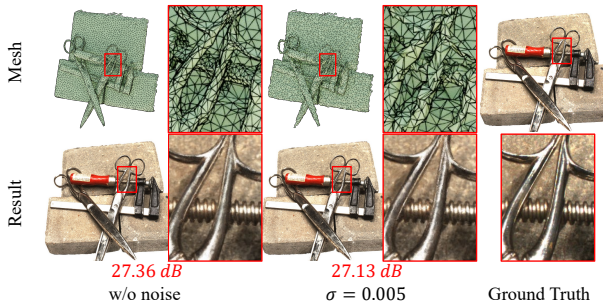

Figure 17: *The top row visualizes the original and noise injected meshes that share the same mesh parameterization. The bottom row shows the rendering results and average PSNR values.*

Figure 18: *Limitation of our method. The thin stem part has been missed in the base mesh. Accordingly, its appearance is not properly reconstructed.*

Comparison on high-gradient regions We propose a novel comparison method that computes the image metrics in high-gradient texture regions in order to measure the reconstruction performance for texture details. We use the canny edge detector [Can86] to find patches with high image gradients. We then divide the image into uniform 2D patches and rank them with respective to the number of edge pixels. Finally, we compute the image metrics on top ranking patches. Our method achieves compelling results quantitatively and qualitatively in high-gradient regions, as shown in Table 1 and Figure 15.

5.3. Discussion

Appearance model compatibility In this paper, we adopt TensoRF [CXG22] for the appearance model trained using the surface-aligned volume. However, since the surface-aligned volume is represented as the axis-aligned uvh-space, we can replace the appearance model by any other volumetric scene representation than TensoRF. Figure 16 demonstrates the compatibility of our method with InstantNGP [MESK22] as well as TensoRF, achieving PSNR values of 27.58 dB and 27.80 dB on the DTU dataset, respectively.

Robustness on noise Our volumetric representation enables robust reconstruction despite a certain level of geometric error in the guide mesh. To check this robustness, we conducted an experiment where our model is trained with a noisy guide mesh. In the experiment, we used the DTU 37 dataset, where the diagonal length of the mesh bounding box is 2.25. We add Gaussian noise to the mesh with a random value from the normal distribution $\mathcal{N}(0, 0.005^2)$. We found that our representation is robust to minor mesh noise both qualitatively and quantitatively, as shown in Figure 17.

Limitation Our surface-aligned volume used for appearance modeling is determined by the guide mesh. Consequently, if some geometry of the scene is missing in the guide mesh, our method fails to reconstruct the appearances of such parts (Figure 18).

6. Conclusion

We presented a mesh-based neural rendering method that reconstructs high-fidelity appearances within a surface-aligned volume surrounding the guide mesh. Our method can handle geometry editing by modifying the guide mesh and texture editing through manipulations on the 2D domain in the surface-aligned volume. We demonstrated high-quality rendering of our method and more flexible and accurate editing operations than the state-of-the-art methods as well as a few applications. An interesting future work would be to extend our approach to cover thin geometric structures that can be hardly captured by reconstruction methods.

Acknowledgements

We thank the anonymous reviewers for the valuable feedback. This work was supported by NRF grants (RS-2023-00280400, RS-2024-00451947, RS-2023-00212828) and IITP grants (ICT Research Center, RS-2024-00437866; AI Innovation Hub, RS-2021-II212068; AI Graduate School Program, RS-2019-II191906) funded by Korea government (MSIT).

References

[BUE01] BUEHLER C.: Unstructured lumigraph rendering. In *Proc. SIGGRAPH'01* (2001), pp. 425–432. 2

[Can86] CANNY J.: A computational approach to edge detection. *IEEE Transactions on pattern analysis and machine intelligence*, 6 (1986), 679–698. 9

[CVM*96] COHEN J., VARSHNEY A., MANOCHA D., TURK G., WEBER H., AGARWAL P., BROOKS F., WRIGHT W.: Simplification envelopes. In *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques* (1996), pp. 119–128. 3

[CXG*22] CHEN A., XU Z., GEIGER A., YU J., SU H.: Tensorf: Tensorial radiance fields. In *Computer Vision–ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part XXXII* (2022), Springer, pp. 333–350. 3, 4, 5, 8, 9

[FKYT*22] FRIDOVICH-KEIL S., YU A., TANCIK M., CHEN Q., RECHT B., KANAZAWA A.: Plenoxels: Radiance fields without neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2022), pp. 5501–5510. 2, 8

[GH97] GARLAND M., HECKBERT P. S.: Surface simplification using quadric error metrics. In *Proceedings of the 24th annual conference on Computer graphics and interactive techniques* (1997), pp. 209–216. 3

[HDD*93] HOPPE H., DEROSE T., DUCHAMP T., MCDONALD J., STUETZLE W.: Mesh optimization. In *Proceedings of the 20th annual conference on Computer graphics and interactive techniques* (1993), pp. 19–26. 3

[IZN*16] INNMANN M., ZOLLHÖFER M., NIESSNER M., THEOBALT C., STAMMINGER M.: Volumedeform: Real-time volumetric non-rigid reconstruction. In *Proceedings of the European Conference on Computer Vision (ECCV)* (2016). 2

[JDV*14] JENSEN R., DAHL A., VOGIATZIS G., TOLA E., AANÆS H.: Large scale multi-view stereopsis evaluation. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (2014), pp. 406–413. 5

[JMD*07] JOSHI P., MEYER M., DEROSE T., GREEN B., SANOCKI T.: Harmonic coordinates for character articulation. *ACM transactions on graphics (TOG) 26*, 3 (2007), 71–es. 5

[JXX*21] JIAKAI Z., XINHANG L., XINYI Y., FUQIANG Z., YANSHUN Z., MINYE W., YINGLIANG Z., LAN X., JINGYI Y.: Editable free-viewpoint video using a layered neural representation. In *ACM SIGGRAPH* (2021). 2

[KKLD23] KERBL B., KOPANAS G., LEIMKÜHLER T., DRETTAKIS G.: 3d gaussian splatting for real-time radiance field rendering. *ACM Transactions on Graphics (ToG) 42*, 4 (2023), 1–14. 2

[KVH84] KAJIYA J. T., VON HERZEN B. P.: Ray tracing volume densities. *ACM SIGGRAPH computer graphics 18*, 3 (1984), 165–174. 4

[LC98] LORENSEN W. E., CLINE H. E.: Marching cubes: A high resolution 3d surface construction algorithm. In *Seminal graphics: pioneering efforts that shaped the field*. 1998, pp. 347–353. 3

[LGTK23] LI R., GAO H., TANCIK M., KANAZAWA A.: Nerfacc: Efficient sampling accelerates nerfs. *arXiv preprint arXiv:2305.04966* (2023). 4

[LH96] LEVOY M., HANRAHAN P.: Light field rendering. Association for Computing Machinery. 2

[LZZ*21] LIU S., ZHANG X., ZHANG Z., ZHANG R., ZHU J.-Y., RUSSELL B.: Editing conditional radiance fields. In *Proceedings of the International Conference on Computer Vision (ICCV)* (2021). 3

[MESK22] MÜLLER T., EVANS A., SCHIED C., KELLER A.: Instant neural graphics primitives with a multiresolution hash encoding. *arXiv preprint arXiv:2201.05989* (2022). 2, 3, 9

[MST*20] MILDENHALL B., SRINIVASAN P. P., TANCIK M., BARRON J. T., RAMAMOORTHI R., NG R.: Nerf: Representing scenes as neural radiance fields for view synthesis. 2, 4, 5, 8

[NFS15] NEWCOMBE R. A., FOX D., SEITZ S. M.: Dynamicfusion: Reconstruction and tracking of non-rigid scenes in real-time. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (June 2015). 2

[Pau98] PAUL D.: Rendering synthetic objects into real scenes: bridging traditional and image-based graphics with global illumination and high dynamic range photography. *Computer Graphics 32* (1998), 189–198. 2

[SA07] SORKINE O., ALEXA M.: As-rigid-as-possible surface modeling. In *Symposium on Geometry processing* (2007), vol. 4, Citeseer, pp. 109–116. 5

[SSC22] SUN C., SUN M., CHEN H.: Direct voxel grid optimization: Super-fast convergence for radiance fields reconstruction. In *CVPR* (2022). 2

[SSP07] SUMNER R. W., SCHMID J., PAULY M.: Embedded deformation for shape manipulation. In *ACM SIGGRAPH 2007 Papers* (New York, NY, USA, 2007), SIGGRAPH '07, Association for Computing Machinery, p. 80–es. URL: https://doi.org/10.1145/1275808.1276478, doi:10.1145/1275808.1276478. 2

[WHH*23] WANG Y., HAN Q., HABERMANN M., DANIILIDIS K., THEOBALT C., LIU L.: Neus2: Fast learning of neural implicit surfaces for multi-view reconstruction. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)* (2023). 3

[WLL*21] WANG P., LIU L., LIU Y., THEOBALT C., KOMURA T., WANG W.: Neus: Learning neural implicit surfaces by volume rendering for multi-view reconstruction. *arXiv preprint arXiv:2106.10689* (2021). 8

[WS23] WEIWEI SUN EDUARD TRULLS Y.-C. T. S. S. G. S. A. T. K. M. Y.: Pointnerf++: A multi-scale, point-based neural radiance field. In *arXiv* (2023). 2

[WSLG23] WU T., SUN J.-M., LAI Y.-K., GAO L.: De-nerf: Decoupled neural radiance fields for view-consistent appearance editing and high-frequency environmental relighting. In *ACM SIGGRAPH* (2023). 3, 8

[WZY*23] WANG X., ZHU J., YE Q., HUO Y., RAN Y., ZHONG Z., CHEN J.: Seal-3d: Interactive pixel-level editing for neural radiance fields. In *Proceedings of the IEEE/CVF International Conference on Computer Vision* (2023), pp. 17683–17693. 3

[XH22] XU T., HARADA T.: Deforming radiance fields with cages. In *ECCV* (2022). 2, 8

[XXH*21] XIANG F., XU Z., HASAN M., HOLD-GEOFFROY Y., SUNKAVALLI K., SU H.: Neutex: Neural texture mapping for volumetric neural rendering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2021), pp. 7119–7128. 2

[XXP*22] XU Q., XU Z., PHILIP J., BI S., SHU Z., SUNKAVALLI K., NEUMANN U.: Point-nerf: Point-based neural radiance fields. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2022), pp. 5438–5448. 2

[YBZ*22] YANG B., BAO C., ZENG J., BAO H., ZHANG Y., CUI Z., ZHANG G.: Neumesh: Learning disentangled neural mesh-based implicit field for geometry and texture editing. In *European Conference on Computer Vision* (2022), Springer, pp. 597–614. 1, 2, 3, 5, 7, 8

[YSL*22] YUAN Y.-J., SUN Y.-T., LAI Y.-K., MA Y., JIA R., GAO L.: Nerf-editing: geometry editing of neural radiance fields. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2022), pp. 18353–18364. 2, 8

[YZX*21] YANG B., ZHANG Y., XU Y., LI Y., ZHOU H., BAO H., ZHANG G., CUI Z.: Learning object-compositional neural radiance field for editable scene rendering. In *International Conference on Computer Vision (ICCV)* (October 2021). 2

[ZSGS04] ZHOU K., SYNDER J., GUO B., SHUM H.-Y.: Iso-charts: stretch-driven mesh parameterization using spectral analysis. In *Proceedings of the 2004 Eurographics/ACM SIGGRAPH symposium on Geometry processing* (2004), pp. 45–54. 3