Interactive NeRF Geometry Editing with Shape Priors

Yu-Jie Yuan, Yang-Tian Sun, Yu-Kun Lai, Yuewen Ma, Rongfei Jia, Leif Kobbelt, and Lin Gao*

Abstract—Neural Radiance Fields (NeRFs) have shown great potential for tasks like novel view synthesis of static 3D scenes. Since NeRFs are trained on a large number of input images, it is not trivial to change their content afterwards. Previous methods to modify NeRFs provide some control but they do not support direct shape deformation which is common for geometry representations like triangle meshes. In this paper, we present a NeRF geometry editing method that first extracts a triangle mesh representation of the geometry inside a NeRF. This mesh can be modified by any 3D modeling tool (we use ARAP mesh deformation). The mesh deformation is then extended into a volume deformation around the shape which establishes a mapping between ray queries to the deformed NeRF and the corresponding queries to the original NeRF. The basic shape editing mechanism is extended towards more powerful and more meaningful editing handles by generating box abstractions of the NeRF shapes which provide an intuitive interface to the user. By additionally assigning semantic labels, we can even identify and combine parts from different objects. We demonstrate the performance and quality of our method in a number of experiments on synthetic data as well as real captured scenes.

Index Terms—Neural Radiance Fields, Geometry Editing, Shape Deformation, Interactive Editing

1 Introduction

Novel view synthesis has been extensively studied in computer vision and computer graphics. In particular, the recently proposed neural radiance field (NeRF) [1] has inspired a large number of follow-up works aiming to achieve better visual effects [2], faster rendering speed [3], [4], generalization to different scenes [5], relighting [6], [7], applying to dynamic scenes [8], and reducing the number of inputs [9], [10]. However, as an implicit modeling method, the neural radiance field is difficult for users to interactively edit or modify the scene objects, which is relatively easy with explicit representations. The mesh representation, as a kind of explicit representation, is commonly used in shape modeling and rendering. There is a lot of research work on mesh deformation or editing [11]. However, it is difficult to obtain an accurate explicit representation of a real-world scene. From a sparse set of images, one can use some Multi-View Stereo (MVS) method [12] to reconstruct the point cloud or mesh representation of the scene, but the quality is generally poor. Rendering the reconstructed representation under novel views often leads to unrealistic results. Therefore, based on the promising novel view synthesis ability of implicit representations such as NeRFs, further studying how to edit the implicit representation has become a new research direction.

There has been some initial work exploring NeRF editing. Given a NeRF to be edited, EditNeRF [13] allows users to draw a few coarse scribbles on a rendered image of the NeRF from a random view, and optimizes the network parameters by back propagation to achieve consistent modification across all views. The network is trained on a set of synthetic data from the same category and relies on the data prior to enable the propagation of user edits from 2D images to 3D space. However, this method is difficult to generalize to real data with complex textures and does not support shape deformation of objects. Zhang et al. [14] propose an editable free-viewpoint video reconstruction method. They split dynamic humans in the scene from the background and use different NeRF networks to model individual people, enabling editing of each person in the time and space domains, such as modifications to the position and size of a person and adjustments to the timing of an action. However, the method cannot modify human poses, or edit objects other than people. Similarly, there is also work on modeling static scenes, where changes to the position or size of objects can be achieved by decoupling the scene [15], or by organizing the scene using an octree [16]. Recently, Kania et al. [17] proposed a controllable neural radiance field that connects the user-annotated mask of attribute regions and attribute values with the deformation of rays, enabling the adjustment of attribute values to change the state of the corresponding attribute regions. This method relies on user annotations and works better on human faces. It also produces blurred rendering after deformation. Meanwhile, some works [8], [18] consider using NeRF to model dynamic scenes and using Multi-Layer Perceptron (MLP) to predict scene changes. However, they either limit the edits to human bodies [19], or can only learn motion

- * Corresponding Author is Lin Gao (gaolin@ict.ac.cn).
- Yu-Jie Yuan, Yang-Tian Sun, and Lin Gao are with the Beijing Key Laboratory of Mobile Computing and Pervasive Device, Institute of Computing Technology, Chinese Academy of Sciences, Beijing, China, and also with the University of Chinese Academy of Sciences, Beijing, China. E-Mail: {yuanyujie, sunyangtian, gaolin}@ict.ac.cn
- Yu-Kun Lai is with the School of Computer Science & Informatics, Cardiff University, U.K. E-mail: LaiY4@cardiff.ac.uk
- Yuewen Ma is with PICO, ByteDance, Beijing, China. E-mail: mayuewen@bytedance.com
- Rongfei Jia is with Beijing ZaoWuHuiJing Technology Co., Ltd, Beijing, China. E-mail: iceprg@126.com
- Leif Kobbelt is with Computer Graphics Group, RWTH Aachen University, Germany. E-mail: kobbelt@cs.rwth-aachen.de

information from the recorded videos, and cannot perform editing [8].

In this paper, we propose an interactive system for real-time geometry-aware shape deformation of neural radiance fields that combines the advantage of explicit representations for easy local editing and the advantage of implicit representations for realistic rendering effects. Our method not only supports user-specified control points to deform the neural radiance field, but more importantly, provides the user with automatically extracted intuitive deformation handles for easy manipulation. Different from the previous work [13], [15], we focus on the geometric content of the scene, as shown in Fig. 1, supporting users to edit the scene geometry, and can perform photo-realistic rendering of the edited scenes from novel views. To this end, we first extract an explicit triangle mesh representation from the trained NeRF. The explicit mesh representation is then intuitively deformed by the user. Next, a tetrahedral mesh is built from the triangle mesh representation, which wraps around the triangle mesh. We use the deformation of the triangle mesh to drive the deformation of the tetrahedral mesh, which propagates the deformation of the scene geometric surface to the spatial discrete deformation field. Finally, we use tetrahedral vertex interpolation to complete the propagation from the discrete deformation field to the continuous deformation field. The rays passing through the tetrahedral mesh will be bent accordingly following the continuous deformation field, so that the final rendering result conforms to the user's edits. Although the user needs to manually specify the control points, our method is general, not limited to specific shapes such as human bodies, and applicable to arbitrary shapes such as animal models and general man-made objects.

To further support geometry-aware editing, we propose to automatically generate box handles for intuitive interaction. The key idea is to introduce geometric properties (such as shape and structure) of the objects in NeRF to the deformation process to aid the user's editing. Specifically, we first extract the explicit geometry representation as a mesh from real-captured images and perform geometric analysis on the extracted mesh, including shape abstraction as a collection of boxes, segmentation, and symmetry detection between box abstractions. The extracted box abstractions will then be used as the deformation handles for the user to edit the geometry of NeRF, and the symmetry information will help the user to synchronize the editing between symmetrical box handles to reduce the interaction burden, i.e., if the user edits one of the box handles, other symmetrical boxes will have a corresponding deformation to maintain the symmetry. Furthermore, based on the segmented parts, our system allows the user to combine semantic parts from different objects to form a new NeRF, as shown in Fig. 2. Our system supports real-time shape deformation editing and free-view rendering of the results. In summary, our work has the following three contributions:

- We propose a system for real-time interactive editing of NeRF geometry, which automatically generates object-aligned boxes as the deformation handles for intuitive editing.
- We propose a geometry-aware approach to deforming neural radiance field geometry, which maintains the geometric properties of the object when manipulating the handles to deform the NeRF geometry.
- Our approach also enables users to combine semantic parts from distinct objects to create new ones.

This paper substantially extends our original conference version [20] in the following ways: We propose a geometry-aware shape editing method for static NeRFs. We build an interactive NeRF-Editing system that can automatically generate boxes as deformation handles for intuitive user interaction, and maintain the geometric property, in particular symmetry, during the editing. We further propose a composition approach based on shape segmentation that supports combining shape parts from different NeRFs into a new NeRF. The code will be publicly released.

2 Related Work

Our NeRF editing framework provides a new paradigm for novel view synthesis of an edited neural implicit scene representation. Here, we summarize related work of novel view synthesis and 3D deformation/editing methods.

Novel view synthesis. To infer the photo-realistic novel view synthesis result from given input images, prior works rely on explicit [21], [22], [23], [24] or implicit [25], [26], [27] geometry representation of the real world scene. Recently, used both as a component in deep neural network pipelines and as a standalone rendering pipeline, neural rendering has achieved immense progress, which is comprehensively summarized in [28], [29]. It adopts deep neural networks to synthesize images, which can be employed on multiple representations, such as voxels [30], [31], point clouds [32], [33], meshes [34], [35], [36], [37], multi-plane images (MPIs) [38], [39], [40] and implicit fields [41], [42]. As one of the representative works, Neural Radiance Field (NeRF) [1] has attracted a lot of attention, which uses a multi-layer perceptron (MLP) to model the geometry and appearance of a scene. NeRF can achieve photo-realistic synthesis of novel view images with view-dependent effects. However, NeRF still has shortcomings and plenty of work has extended the original NeRF, including better synthesis effects [2], [43], [44], applicable to dynamic scenes [8], [14], [18], [19], [45], [46], [47], [48], [49], faster training and rendering speed [3], [4], [50], [51], [52], generalization to different scenes [53], [54], relighting [6], [7], [55], [56], and various kinds of editing [20], [57], [58], [59], [60], [61]. In this work, we focus on geometry editing/deformation for NeRF. Most of the existing NeRF editing work adopts the idea of decoupling, such as decoupling geometry and appearance [13], decomposing scenes into individual objects and the background [14], [15], and disentangling images into different attribute areas [17]. CoNeRF [17] can modify the object shape by changing the attribute values. The method is based on dynamic videos, and users are required to label attribute values and corresponding attribute areas, so only states that have appeared in the video can be generated, while unseen states cannot be generated. Our framework instead supports editing the geometric shape of the objects in a static NeRF, which can then be used to synthesize photo-realistic novel view images for visualization. More recently, after the publication of our conference version [20], NeuMesh [62] proposed to define

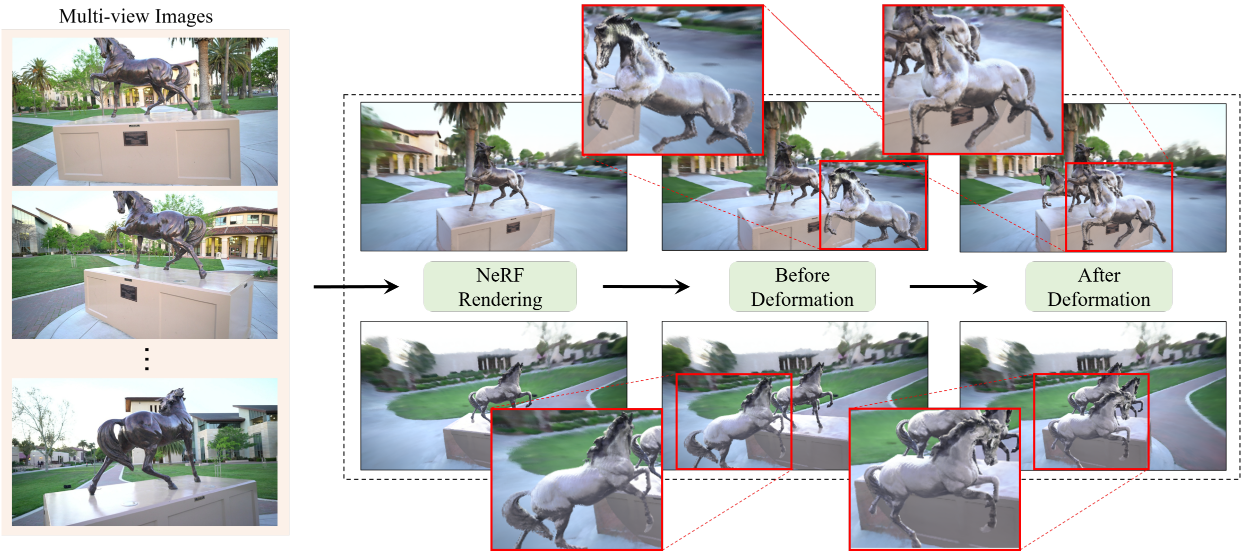


Figure 1. We propose a method to edit a static neural radiance field (NeRF). Users only need to capture multi-view images to build a NeRF representation, and then they can explicitly and intuitively edit the implicit representation of the scene. Our method can perform user-controlled shape deformation on the geometry of the scene, which contains multiple objects.

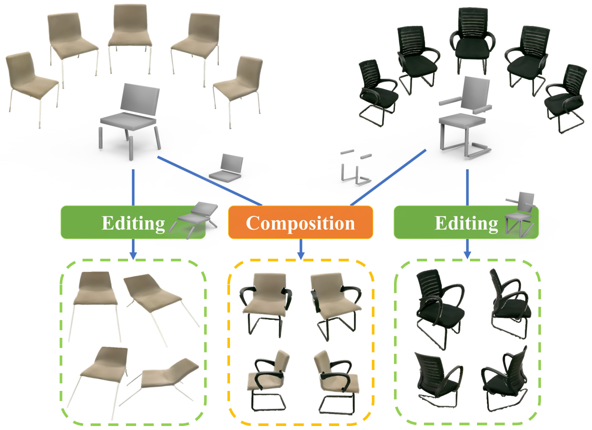


Figure 2. We propose a system for interactive editing of neural radiance fields (NeRFs), which can deform the NeRF geometry. The deformation is geometry-aware and controlled by some automatically extracted box abstractions as interaction handles. Our method also supports recombination of parts from different models to form new ones, which also enables free view synthesis.

geometric features and appearance features on mesh vertices, which are then combined with a neural radiance field for rendering. The method can achieve geometric editing and appearance editing. There has also been work [63] considering the use of tetrahedral deformation to bend rays to achieve geometric deformation of NeRF. However, these works do not consider the preservation of geometric properties and the provision of intuitive handles. Our method is able to automatically generate boxes as deformation handles for user interaction, and maintain the geometric properties, in particular symmetry, during editing. Our method also supports part-level NeRF compositions.

3D deformation and editing methods. Editing a 3D model means deforming the shape of the model under some controls given by the user. There has been much work about the editing of explicit geometry representations [64], [65], which we refer readers to a recent survey [11]. Traditional mesh deformation methods are based on Laplacian coordinates [66], [67], [68], Poisson equation [69], and dual Laplacian coordinates [70]. As a representative work among them, ARAP (As-Rigid-As-Possible) deformation [71] is an interactive mesh editing scheme, which preserves details during the deformation by maintaining the rigidity of local transformations. Another approach to driving mesh deformation is through a proxy, such as skeletons [72], [73] or cages [74], [75], [76]. These methods need to calculate the weights [77], [78], [79] between the proxy and the mesh vertices, and propagate the transformation of the proxy to the mesh. With the proliferation of geometric models [80], data-driven deformation [81], [82], [83] becomes available, which analyzes the deformation prior of existing shapes in the dataset and produces more realistic results. At the same time, plenty of data also allows neural networks to be introduced into 3D editing [84], [85], [86], [87]. In addition to explicit mesh representations, implicit fields can also be edited in combination with a neural network. Deng et al. proposed the deformed implicit field [88], which is capable of modeling dense surface correspondence and shape editing based on the learned information from an object category. Our work also aims to edit implicit representations, in particular NeRFs. The difference is that we take advantage of the intuitive and convenient characteristics of explicit mesh editing. By establishing a connection between the explicit mesh representation and implicit neural representation, well-developed mesh deformation methods can be used to edit the geometry of the implicit representation. In addition to considering control points as deformation handles, we propose to automatically generate box abstractions as the handles for editing, and we also take into account geometric property preservation of the model during editing.

3 Our Method

Our work is based on the neural radiance field (NeRF) [1], which has promising performance in novel view synthesis. As a result, our method enables users to interactively edit the content of the scene in a geometry-aware manner, and can generate new images from arbitrary views after editing.

We will first briefly review NeRF pipeline (Sec. 3.1), and then introduce how to extract the explicit triangle mesh representation from the implicit representation of the scene and enable users to edit the mesh representation (Sec. 3.2). After the user edits the triangle mesh representation of the scene, we need to transfer this deformation to the implicit volume representation. We split the transfer into two steps. The first step is to transfer the surface mesh deformation to a volumetric mesh, where we build a tetrahedral mesh that wraps around the surface mesh, and transfer user edits on the surface mesh to a discrete deformation field on the tetrahedral mesh (Sec. 3.3). The next step is to transform the discrete deformation field to a continuous deformation field in the volume space, which is used to guide the bending of the rays to render images conforming to user edits (Sec. 3.4). Later, in Sec. 4.3, we will show that directly transferring the deformation from the surface mesh to the implicit volume by interpolation will lead to visible artifacts compared to our two step strategy. We also propose an interactive geometry-aware NeRF editing system based on the above method. We will describe how to perform geometric analysis of objects modeled by the NeRF, including shape abstraction, segmentation and symmetry detection (Sec. 3.5). Using the control of the box abstractions, the NeRF geometry is deformed under the constraints of symmetry and segmentation information (Sec. 3.6). Our system further supports the re-combination of segmented parts from different NeRFs into a new NeRF. Finally, we will present our real-time interactive editing system (Sec. 3.7). Our method establishes the connection between the explicit mesh representation and the implicit radiance field, enabling users to modify the geometry of radiance field through intuitive edits. The system overview is shown in Fig. 3.

3.1 Neural Radiance Fields

Neural Radiance Field or NeRF [1] proposes to use a multi-layer perceptron (MLP) network to model the geometry and appearance of the scene from a sparse set of images. Given the known camera parameters, the image pixels can be transformed to the world coordinate system and connected with the camera position to generate the light rays that are directed toward the scene. NeRF samples points on the ray and uses volume rendering [89] to composite the color of each ray. The spatial coordinates $\mathbf{p} = (x, y, z)$ of each sampled point and the ray direction $\mathbf{d} = (\theta, \phi)$ will go through positional encoding $\zeta(\cdot)$, and then be passed to the fully connected network to predict the volume density σ and RGB value $\mathbf{c}$: $F_\Theta : (\zeta(\mathbf{p}), \zeta(\mathbf{d})) \rightarrow (\sigma, \mathbf{c})$, where Θ represents the network weights. The predicted density value σ can be interpreted as the differentiable probability of the ray terminated at the sampled point, and the color $\hat{C}(\mathbf{r})$ of the image pixel corresponding to the ray $\mathbf{r}(t)$ can be calculated through discrete accumulation:

$$\hat{C}(\mathbf{r}) = \sum_{i=1}^{N} \exp(-\sum_{j=1}^{i-1} \sigma_j \delta_j)(1 - \exp(-\sigma_i \delta_i))\mathbf{c}_i, \quad (1)$$

where $\delta_i = t_{i+1} - t_i$ is the distance between successive samples. The network is supervised by the RGB loss function, which is calculated between the generated color $\hat{C}(\mathbf{r})$ and the ground truth color $C(\mathbf{r})$ of the ray.

3.2 Editing of Explicit Surface Mesh Representation

After the NeRF network is trained, an explicit triangle mesh representation can be extracted directly from the neural radiance fields using Marching Cubes [90]. However, the mesh extracted from the original NeRF network is often of low quality. In order to obtain a satisfactory editing representation, we adopt the reconstruction method proposed in NeuS [91], which takes a bias-free volume rendering approach to learn the geometry as a neural signed distance function (SDF) representation. The mesh representation extracted from the zero-level set of SDF will serve as the user's editing proxy, allowing users to edit the scene content intuitively. In this paper, we use the classic ARAP (as-rigid-as-possible) deformation method [71] to enable users to interactively deform the mesh. It should be noted that any other mesh deformation method can be used here, including skeleton-based and cage-based methods.

The extracted triangle mesh is denoted as S, and $N(i)$ represents the index set of vertices adjacent to vertex i. We further denote $\boldsymbol{v}_i \in \mathbb{R}^3$ as the position of the vertex i on the mesh S. After the user's edits, the mesh S is transformed to the deformed mesh S' with the same connectivity and different vertex positions $\boldsymbol{v}'_i$, treating user editing as boundary conditions. The overall ARAP deformation energy is to measure the rigidity of the entire mesh and is the sum of the distortion energies of each deformation cell, including vertex i and its 1-ring neighbors, shown in Eq. 2.

$$E(S') = \sum_{i=1}^{n} \sum_{j \in N(i)} w_{ij} \| (\boldsymbol{v}'_i - \boldsymbol{v}'_j) - \mathbf{R}_i (\boldsymbol{v}_i - \boldsymbol{v}_j) \|^2. \quad (2)$$

Here, $w_{ij} = \frac{1}{2}(\cot \alpha_{ij} + \cot \beta_{ij})$ is the cotangent weight, and α_{ij}, β_{ij} are the angles opposite to the mesh edge (i, j). $\mathbf{R}_i$ is the local rotation matrix at vertex i. The deformed shape S' is obtained by minimizing the ARAP energy, which can be efficiently solved by alternately optimizing local rotations $\mathbf{R}_i$ and deformed positions $\boldsymbol{v}'_i$. We refer the readers to [71] for the specific optimization process.

3.3 Deformation Transfer to Discrete Volume

After the user edits the triangle mesh representation of the scene, the deformation needs to be transferred to the implicit volume representation. As introduced before, we split the transfer into two steps. In the first step, we build a tetrahedral mesh (a discrete volumetric representation) to cover the extracted triangle mesh. Starting from the extracted triangle mesh S, we first calculate a cage mesh that encloses the mesh S. This can be achieved by enlarging the triangle mesh by shifting all the vertices by a certain distance offset from the mesh surface in normal direction. We set the default value to 5% of the average distance from the camera positions to object center. The interior of the cage mesh can be regarded as the "effective space" of the implicit volume, because the area near real geometry surface of the scene is enclosed by this cage mesh. When editing larger scenes with multiple objects, this design also ensures other objects not being edited are not affected. We use the tetrahedralization method, TetWild [92], to tessellate the cage mesh to obtain a tetrahedral mesh representation T. It should be noted that the extracted triangle mesh S

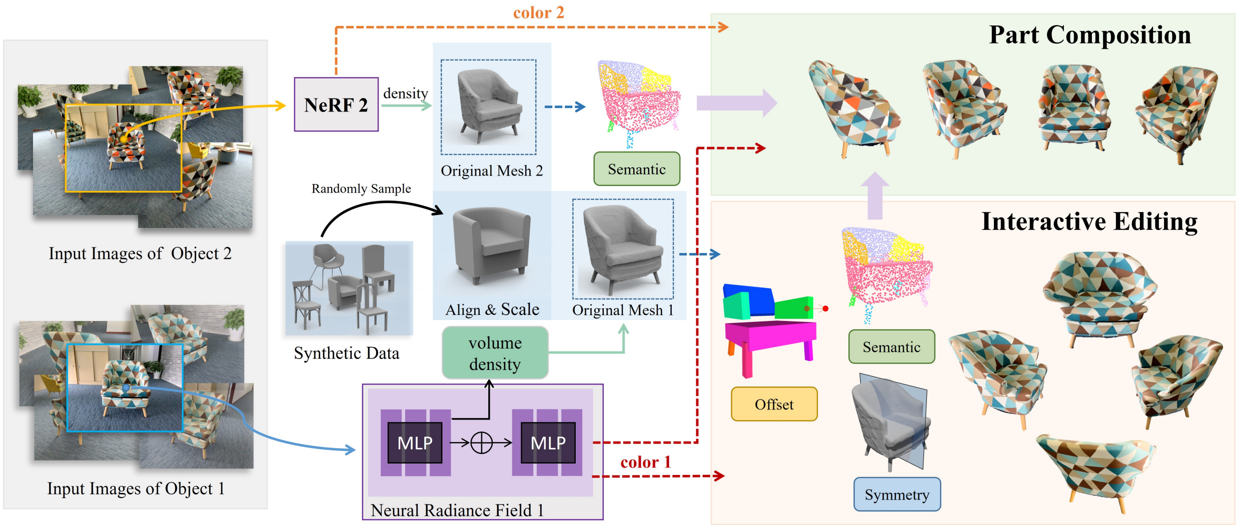


Figure 3. The pipeline of our geometry-aware NeRF editing framework. Leveraging the shape priors brought in by synthetic data, our method enables geometric analysis of real reconstructed models and interactive geometric editing utilizing the box abstractions as interaction handles. At the same time, the segmentation enables our method to re-combine parts from different models to form new ones.

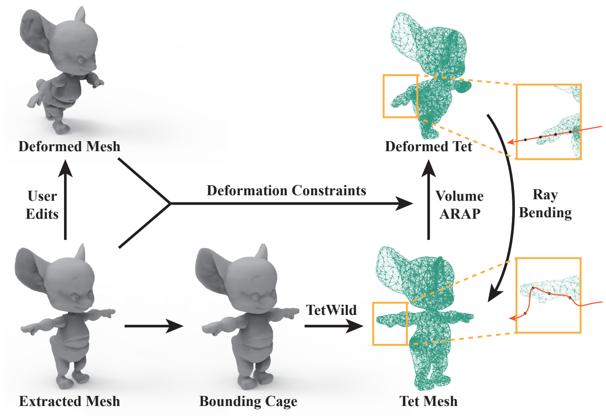


Figure 4. We use deformations specified by users to bend the rays.

is also wrapped in the tetrahedral mesh T. We visualize some extracted triangle mesh S and the corresponding tetrahedral mesh T in the supplementary material. We use the displacement of the triangle mesh vertices $\boldsymbol{v}_i$ to drive the deformation of the tetrahedral mesh T, which transfers the surface deformation to the tetrahedral mesh. The deformed tetrahedral mesh is denoted as T', and $\boldsymbol{t}_k$ and $\boldsymbol{t}'_k$ denote the vertices of the tetrahedral mesh before and after deformation respectively, where k is the vertex index. Here, we also use the ARAP deformation method to deform the tetrahedral mesh T under the constraints of the surface mesh deformation. Eq. 2 can be extended from the triangle mesh to the tetrahedral mesh straightforwardly. The only difference is that the constraints are changed from user-specified control points to the triangle mesh vertices. We can find which tetrahedron each triangle mesh vertex is located in, and calculate its barycentric coordinates relative to the four vertices of the tetrahedron. Then, the optimization problem is,

$$\min E(T'), \text{ subject to } \boldsymbol{A}\boldsymbol{t}' = \boldsymbol{v}', \tag{3}$$

where $\boldsymbol{A}$ is the barycentric weight matrix. This optimization problem can be converted into linear equations using the Lagrangian multiplier method. Please refer to the supplementary material for the specific derivation.

3.4 Ray Bending

After transferring the surface deformation to the tetrahedral mesh, we can obtain the piece-wise linear deformation field of the "effective space". We now utilize these discrete transformations to bend the casting rays. To generate an image of the deformed radiance field, we cast rays to the space containing the deformed tetrahedral mesh. For each sampled point on the ray, we find which tetrahedron of the deformed tetrahedral mesh T' it is located in. Using the correspondence between T and T', the displacement from the vertices after deformation to the vertices before deformation can be obtained. Through barycentric interpolation of the displacements of the four vertices of the tetrahedron where the sampled point is located, the displacement of the sampled point back to the original "effective space" Δp can be obtained. We add the displacement Δp to the input coordinates of the sampled point to predict the density and RGB values. The ray direction will also be recalculated according to the adjacent sampled points after transformation, which denoted as $\mathbf{d}'$, then

$$(\zeta(\mathbf{p} + \Delta\mathbf{p}), \zeta(\mathbf{d}')) \rightarrow (\sigma, \mathbf{c}). \tag{4}$$

The density and RGB values of the sampled points along the ray are used to calculate the corresponding pixel color using Eq. 1. It should be noted that the sampled points that are not within the tetrahedral mesh T' will not be moved, i.e., the part of the ray outside the tetrahedral mesh will not be bent. The process of building deformation field is illustrated in Fig. 4.

3.5 Geometry Analysis using Shape Priors

The aforementioned ARAP method needs to manually select control points for deformation. Although using control points has high degrees of flexibility, the selection of control points and the deformation operation can still be tricky for inexperienced users. Moreover, the deformation does not take into account the geometric properties of the object, such as symmetry. We now consider how to extract the geometric properties of the reconstructed model and generate the box abstractions that are used as the deformation handles. Note that this extended approach only works for geometry-aware editing (preserving geometry properties during editing) and for objects where shape priors from synthetic data are available, which reduces the difficulty of user operations. Otherwise, the previously described approach still applies.

A single model has limited semantic information, so it is difficult to reliably perform automatic geometry analysis directly on the single reconstructed model. We instead rely on shape priors from the synthetic model dataset to help with the geometric analysis of the real reconstructed model. However, the models in the synthetic dataset have been aligned and have a uniform scale, and different reconstructed models have different orientations and scales. So we first align and scale the reconstructed mesh to match those models in the synthetic dataset. The differences between the reconstructed and synthetic models are typically large, so we adopt a heuristic algorithm, including exhaustive search and ICP (Iterative Closest Point) algorithm [93]. Concretely, we rotate the reconstructed model by 10 degrees along each of the three axes at a time. We then randomly pick a model from the synthetic dataset and scale it to a similar size to the rotated real model. We calculate the Chamfer distance between the rotated real model and the scaled synthetic model and select the rotation angle pair with the smallest distance. We further subdivide the angle near the selected rotation angle and repeat the above operations. After a certain number of repetitions (3 in our method), the ICP algorithm completes the final alignment and the aligned real model will be scaled to match the original synthetic model.

After alignment and scaling, the reconstructed model can be directly input into the network of [94] to extract box abstractions and part segmentation at the same time. Further, we perform global reflection symmetry detection [95] to identify box abstractions that are symmetric to each other with respect to the detected symmetry plane. However, the box pair that should be symmetric may be inconsistent in size, leading to misjudgment as asymmetric. So we use the segmented parts to help judge the symmetric boxes, i.e., if the segmented parts are symmetric, their corresponding box abstractions are also considered symmetric. So far, we obtain the box abstractions, part segmentation, and symmetry pairs. Next, we will use such rich geometric information to perform geometry-aware NeRF editing.

3.6 Geometry-aware NeRF Editing

Based on the extracted geometric information, our method enables interactive editing of NeRF geometry using box abstractions as handles. The symmetry and segmentation information are introduced into the editing at the same time, which can maintain the shape symmetry during editing and the editing will not extend beyond the corresponding semantic part (i.e., not accidentally affecting neighboring parts). Finally, using semantic segmentation, we can also combine parts from different NeRFs into a new model, which naturally supports free-view synthesis.

Box-driven Deformation. Compared with the oriented bounding box (OBB) of the semantic part (that is a box containing all points of the part and usually larger than the box abstraction), the box abstraction is tighter and also reflects the structure of the model, which is beneficial for interaction. Therefore, we choose to use the box abstraction as the interactive deformation handle. First, the box abstractions are edited to drive the deformation of the reconstructed mesh. The user can perform operations including rotation, translation, and scaling on the box. We use $\hat{\mathbf{R}}$, $\hat{\mathbf{t}}$, $\hat{\mathbf{S}}$ to denote the rotation matrix, the translation vector, and the scaling matrix of the box, respectively. These edits are further propagated to the mesh. To achieve this, we find the two closest boxes for each vertex of the mesh, and the weighted combination of the transformations of these two boxes is used as the transformation of the vertex, where the weights are calculated based on the inverse of the distance between the vertex and the box surface. To introduce semantic information into the editing, we first associate semantic segmentation with box handles, where each box handle corresponds to a semantic part. Because semantic segmentation is performed on the point cloud sampled from the mesh surface, for each vertex on the mesh, we find the nearest sampled point and associate the corresponding box (referred as semantic box) of the sampled point with the mesh vertex. Then the transformation of the semantic box is also propagated to the transformation of the mesh vertex. Specifically, in the above weighted combination, the transformation of the semantic box will be given a greater weight, and combined with the transformations of the other two nearest box handles to obtain the transformation of the mesh vertex. Formally, given the rotation matrices $\hat{\mathbf{R}}_1$, $\hat{\mathbf{R}}_2$ of the two closest boxes and the rotation matrix $\hat{\mathbf{R}}_{sem}$ of the semantic box, $dist_1$, $dist_2$ are the distances from the mesh vertex v to the two closest boxes, the distance to the semantic box $dist_{sem}$ is set to be $10^{-5} \times \min(dist_1, dist_2)$, which tends to be much smaller than $dist_1$ and $dist_2$, then the rotation matrix $\hat{\mathbf{R}}_v$ of the vertex v is calculated as

$$\hat{\mathbf{R}}_v = \exp(\log(\hat{\mathbf{R}}_1) \cdot w_1 + \log(\hat{\mathbf{R}}_2) \cdot w_2 + \log(\hat{\mathbf{R}}_{sem}) \cdot w_{sem}), \tag{5}$$

where

$$\begin{aligned} w_1 &= \frac{1/dist_1}{1/dist_1 + 1/dist_2 + 1/dist_{sem}}, \\ w_2 &= \frac{1/dist_2}{1/dist_1 + 1/dist_2 + 1/dist_{sem}}, \\ w_{sem} &= \frac{1/dist_{sem}}{1/dist_1 + 1/dist_2 + 1/dist_{sem}}. \end{aligned}$$

Here, $\log$ is the matrix logarithm, and $\exp$ is the exponential function of the matrix. The use of matrix logarithm and exponential ensures rotation matrices are combined in a meaningful way [96]. And given the translation vectors $\hat{\mathbf{t}}_1$, $\hat{\mathbf{t}}_2$ of the two closest boxes (excluding the semantic box), the translation vector $\hat{\mathbf{t}}_{sem}$ of the semantic box, the translation vector $\hat{\mathbf{t}}_v$ of the vertex v is calculated as

$$\hat{\mathbf{t}}_v = \hat{\mathbf{t}}_1 \cdot w_1 + \hat{\mathbf{t}}_2 \cdot w_2 + \hat{\mathbf{t}}_{sem} \cdot w_{sem}, \tag{6}$$

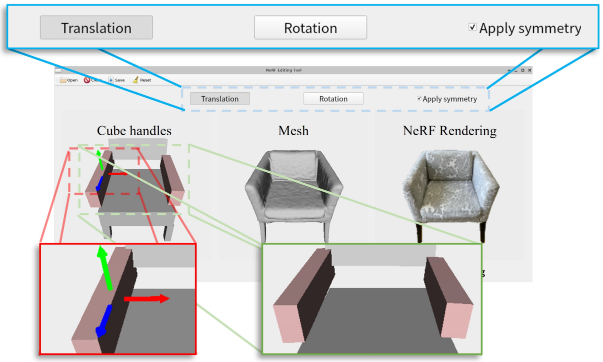


Figure 5. Illustration of our interactive NeRF editing system. It has three subviews that display the box handles, the triangle mesh representation and the NeRF rendering, respectively.

where the calculation of w_1, w_2 and w_{sem} is the same as above. The calculation of the scaling matrix is similar to that of the translation vector.

Further, to maintain the model symmetry and reduce the burden of interaction on the user during the editing, we mark the symmetric box pairs and the user can choose whether to ensure symmetry during editing. If symmetry is maintained, the boxes that are symmetric to the user-edited box will be transformed accordingly at the same time, maintaining the symmetry of the model. After propagating the edits to the mesh, the deformation will be propagated from the mesh to the implicit spatial field in two steps using the method proposed in previous subsections 3.3, 3.4 for NeRF geometry editing.

Part Composition. In addition to interactive geometry-aware shape deformation of NeRF, our system also supports semantic part-level NeRF composition. This relies on the part segmentation extracted by the aforementioned method. Because the box handle corresponds to the segmented part, we can specify the parts to be combined by marking the box handles. It is important to note that since we align the reconstructed real models to match the synthetic data, the inconsistencies in the spatial location and size of different reconstructed models are largely eliminated. However, there are still some cases of misalignment between the parts during combination, especially models from different categories, and the user can adjust the location and size using the box handles. These interactions are not complicated. When the combined NeRF is rendered, each sampled point will obtain its density and color value from the NeRF network corresponding to the OBB (not the box handle as it may not cover the entire part) of the segmented part in which it is located, and finally the colors of all sampled points on each ray will be aggregated through volume rendering. In order to obtain a continuous transition at the boundary of the part combination, we first fuse the segmented point cloud from different models. Then, for each sampled point, we find several (4 in our implementation) nearest points on the fused point cloud. If these closest points are from different NeRF models, then we fuse the colors from different NeRF models based on the inverse distance weighting.

3.7 Interactive NeRF Editing System

Our goal is to provide users with a real-time interactive NeRF editing system, but the rendering speed of the original NeRF cannot support real-time editing. For this reason, we adopt instant-ngp (instant neural graphics primitives) [50] that speeds up NeRF training and rendering. See [50] for details. Since our method will also produce deformed mesh results during editing and combined OBB in the composition, we only sample points near the mesh surface or inside the OBB, which greatly reduces the number of sampled points, reduces the number of times for network inference, and further speeds up rendering. Our system has a total of three subviews that present the interactive deformation handles (i.e. the box abstractions), the reconstructed triangle mesh, and the NeRF rendering result under the corresponding viewpoint, respectively, as shown in Fig. 5. The user can select one of the boxes by clicking it, and rotate, translate and scale each box. The triangle mesh model is driven to deform in real-time, while the edited rendering is also shown. The user changes the viewpoint of the rendering result by changing the viewpoint of the box abstraction. The whole procedure is operated in real-time and the demo of interactive geometry-aware NeRF editing can be found in the supplementary material.

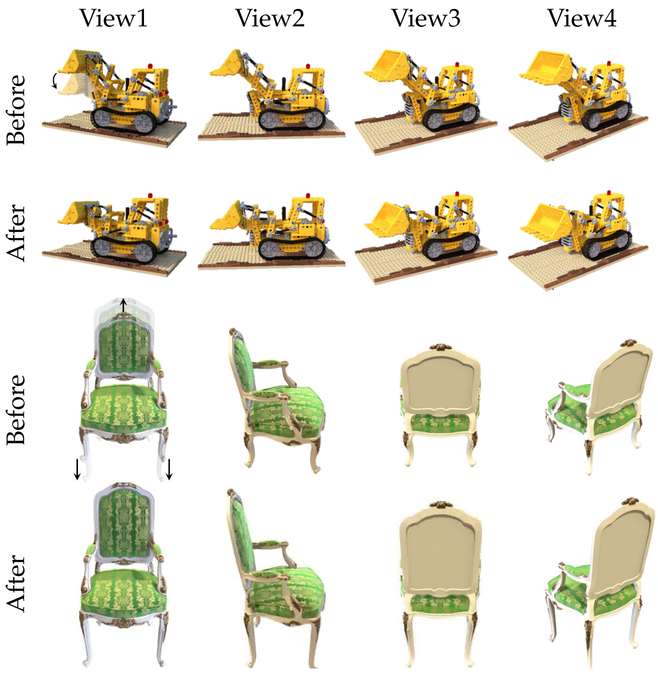


Figure 6. We show the editing results (row “After”) compared with NeRF rendering results (row “Before”) on synthetic data under different views, including a Lego bulldozer and a chair, which illustrate that our method can edit the NeRF of general models. Different columns show different views.

4 Experiments and Evaluations

In this section, we conduct several qualitative and quantitative experiments, including showing editing results on both synthetic data and captured real scenes, comparisons with baseline methods, and ablation study.

4.1 Datasets and metrics

We demonstrate our method on several public synthetic datasets, including some characters in the mixamo [97], the Lego bulldozer and chair from the original NeRF paper [1].

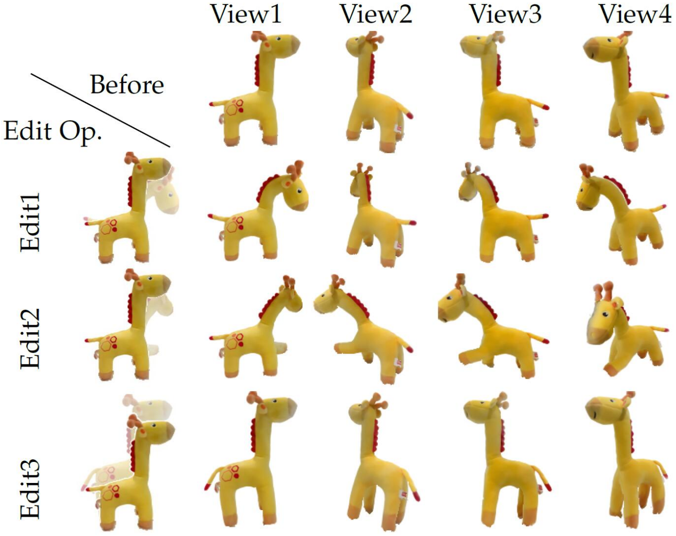


Figure 7. Results of our NeRF editing framework (rows “Edit1”, “Edit2”, “Edit3”) compared with original NeRF results (row “Before”) on a real captured giraffe soft toy. Different columns show different views. We can edit the object into different poses and render it under different views. “Edit Op.” denotes “ Edit Operation”.

Moreover, we also test our method on a real captured horse statue from FVS dataset [36] and several real scenes captured by ourselves. The characters in the mixamo are rendered by ourselves. We generate 100 random views from the upper hemisphere with Blender for training. For the data from NeRF datasets, we use the default training setting of the datasets. For the real scenes captured by ourselves, we leave one image for validation, and the other images are used for training. More information about the self-captured dataset is included in the supplementary document.

It needs to be noted that different from dynamic NeRF methods [8], it is difficult to obtain the ground truths of the novel view synthesis results after user editing, especially on real scenes, as such edited scenes do not physically exist. So we mainly evaluate our approach quantitatively and qualitatively on the characters in the mixamo. Specifically, we rig the mixamo character model, render the deformed characters as the ground truths and compare them with the outputs of our NeRF editing method. We use Structural Similarity Index Measure (SSIM) [98], Learned Perceptual Image Patch Similarity (LPIPS) [99] and Peak Signal-to-Noise Ratio (PSNR) as the metrics to evaluate the performance of our approach. We also evaluate the Fréchet Inception Distance [100] (FID) scores on real scenes to measure the distribution similarity between the results before editing and after editing, since the ground truths are not available.

4.2 Editing Results

Shape editing results under different views. We first show NeRF editing results with manually specified control points rendered from different views in Figs. 6- 8 for synthetic data and real captured objects. For comparison, we also show the results under the same views before editing. In Fig. 6, the first set is a Lego bulldozer from the NeRF dataset. We put down its shovel and achieve the editing of complex synthetic data. The second set is a synthetic chair from the NeRF dataset. We stretch the back and legs of the chair, which demonstrates that our method can edit the local parts of man-made objects. In Fig. 7, we present the editing results on a giraffe soft toy captured by ourselves. It can be seen that users can edit the giraffe to different poses, as well as scale local areas, which demonstrates the usability of our method. In Fig. 8, we show four more sets of results from real scenes to illustrate that our method can be applied to different objects. The wings of the toy dragon are deformed to make them spread out. This can further realize the animation of the dragon flapping its wings while viewing it from different directions. We also show an example of a horse statue from the FVS dataset, where we can deform the horse’s head and raise its hoof. On the example of a laptop, we can rotate its panel to be at different angles. For the real captured chair, we bend the legs of the chair to present another design style, and at the same time enlarge the backrest, which makes the chair more comfortable to sit on. These results show that our approach is able to deform static neural radiance fields according to the user’s editing. In Fig. 1, we show an example of shape deformation for multiple objects in a scene. We first split the mesh of the horse statue from the scene, then copy it into multiple ones, place them in different locations, and deform them differently.

Interactive shape editing with box handles. Our method can also automatically generate box abstractions as deformation handles, and use the box handles to deform the NeRF geometry. We show such interactive editing results with box handles on real captured objects in Figs. 9 and 10. In order to better show the editing effects for comparison, we also show the rendering results before editing under the same views. In Fig. 9, we present the editing results on a typical category, namely chairs. Three different styles of chairs are edited using the box handles. We use symmetry and semantic constraints here. It can be seen that by using the box rotation, we can easily change the chair legs from the vertical state to other states to create different styles of chairs, such as the results of the third group. Boxes can also be translated, changing the position of parts such as the chair legs in the first group, the beams in the second one, and the chair back in the third one. Finally, the use of scaling allows the parts to change size, such as the chair back in the second group, which is enlarged to make people more comfortable to lean on. In Fig. 10, we show more editing results on other categories, such as animal and table. It should be noted that we turn off the symmetry constraint and keep the semantic constraint during the editing. It can be seen that the user can deform the standing giraffe into a running posture, and at the same time can change the direction of the table legs.

Deformation transfer results. In addition to interactive shape deformation, we can also use deformation transfer methods to transfer the deformations from the existing deformation sequence to the objects represented by NeRF. This can achieve some interesting applications. For example, we can transfer the motions of a running horse mesh model [101] to an elk NeRF model. The results are shown in Fig. 11. More cases are shown in the supplementary material where we transfer the movements of a human face from a video clip to a head sculpture and the motions between two human-like characters.

Part composition results. Utilizing shape segmentation

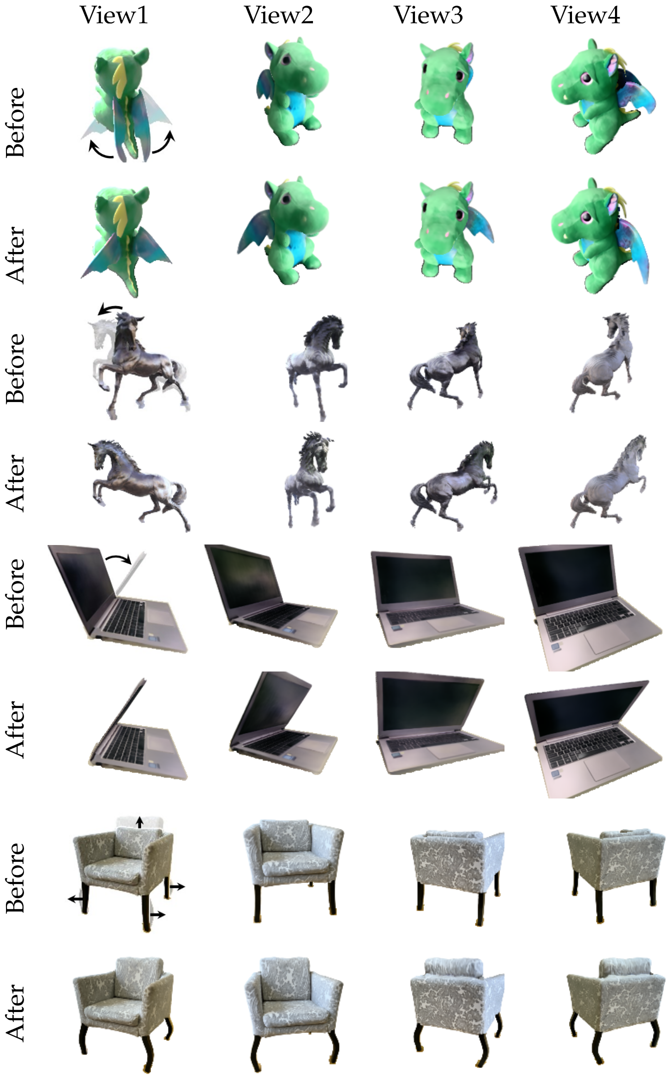


Figure 8. Results of our NeRF editing framework (row "After") compared with original NeRF results (row "Before") on multiple real captured data. Different columns show different views. We edit the static neural radiance fields and exhibit the deformed results under different views.

information, our method also allows the user to combine parts from different NeRFs into a new NeRF and enables free view synthesis of the combined object. We have shown two sets of results in Figs. 2 and 3. Fig. 12 shows two groups of the part composition results from two chairs through part swapping. We also show a combination between the objects from different categories, an elk and a table, in Fig. 13, where the elk legs are combined with the tabletop to form a table supported by the elk legs. From these results, we can see that by using parts from different NeRFs, users can generate new models and can view them from any view.

4.3 Comparisons

As we are the first to perform general geometric shape deformation on NeRF, we propose three baseline methods for comparison. We adopt a naive way for the first baseline to build the correspondence between the extracted triangle mesh and continuous volume space. We no longer construct a tetrahedral mesh and use it as a proxy, but instead directly find the closest point of the sampled point on the extracted triangle mesh surface, and use the displacement of the closest point as the displacement of the sampled point. We denote the first one as "Closest Point". The second baseline is similar to the first one. The difference is that we linearly interpolate the displacements of three nearest triangle mesh vertices, with the coefficients inversely related to distances, to obtain the displacement of the sampled point. We denote this one as "3NN". The last baseline is mesh rendering. The extracted triangle mesh is with vertex color information, which can be directly rendered after user-controlled shape deformation or deformation transfer.

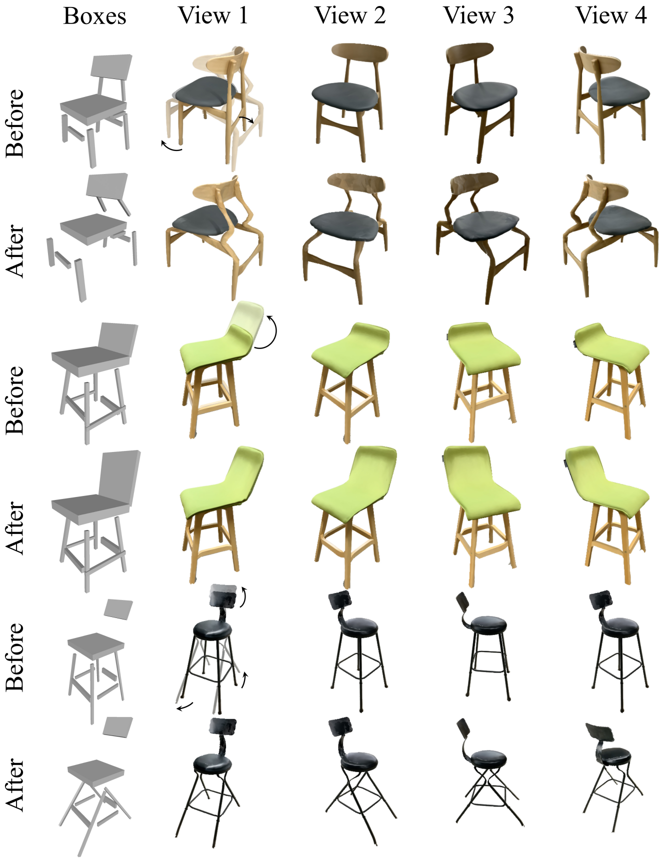


Figure 9. Results of our geometry-aware NeRF editing (row "After") compared with original NeRF results (row "Before") on multiple real captured chairs. We visualize box abstractions in the first column, while other columns show different views. We edit the static neural radiance fields and exhibit the deformed results under different views. Note that we use symmetry information during editing.

We compare our method with the "Closest Point" and "3NN" baselines on the synthetic data mixamo which has ground truth edited results. The visual comparison results are shown in Fig. 14, and the quantitative comparison is shown in Table 1. We also show a visual comparison on a real captured scene in the last row of Fig. 14. Due to the absence of ground truth, we visualize the NeRF rendering result before deformation and corresponding deformation results. It can be seen that the "Closest Point" and "3NN" baselines may cause discontinuities, as they inherit the non-smoothness (of the normal field) of the mesh proxy, so the rendering results have obvious artifacts. While our method adopts two-step deformation transfer, which extends the ARAP deformation to a tetrahedron-envelop with a regularizing effect, and the interpolation in tetrahedron ensures

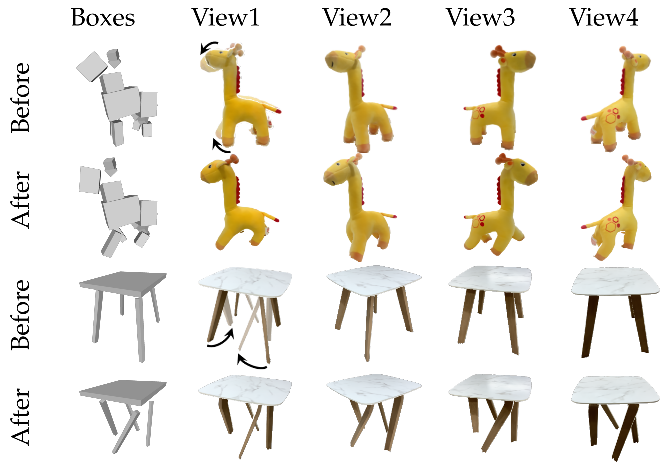


Figure 10. Results of our geometry-aware NeRF editing (row “After”) compared with original NeRF results (row “Before”) on more real captured data. We visualize box abstractions in the first column, while other columns show different views. We edit the static neural radiance fields and exhibit the deformed results under different views. Note that we do not use symmetry information during editing.

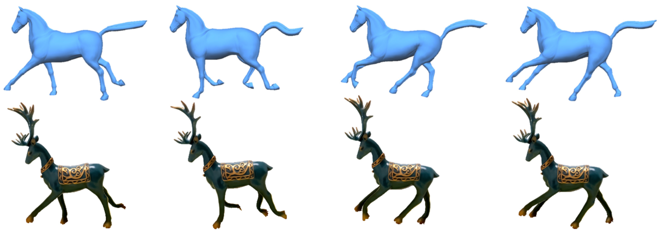


Figure 11. Results of deformation transfer between a horse mesh model and an elk NeRF model. With the help of mesh deformation transfer methods, we can animate the static NeRF model.

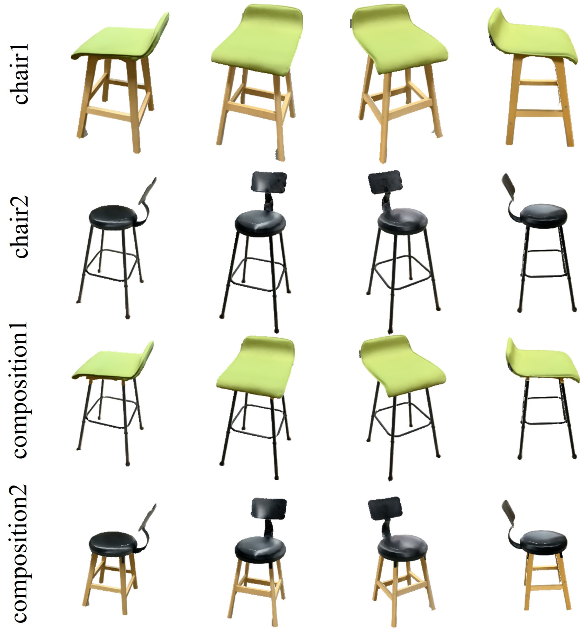


Figure 12. Results of part composition. We exchange the upper and lower parts of two chairs and obtain two composition results.

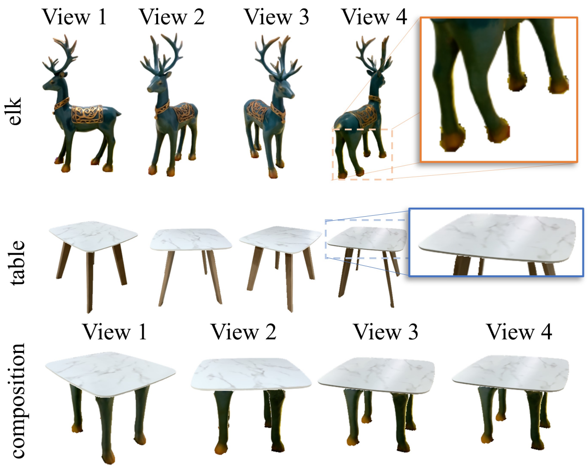


Figure 13. Results of part composition from two objects from different categories. We combine the elk legs and a tabletop to form a table supported by the elk legs.

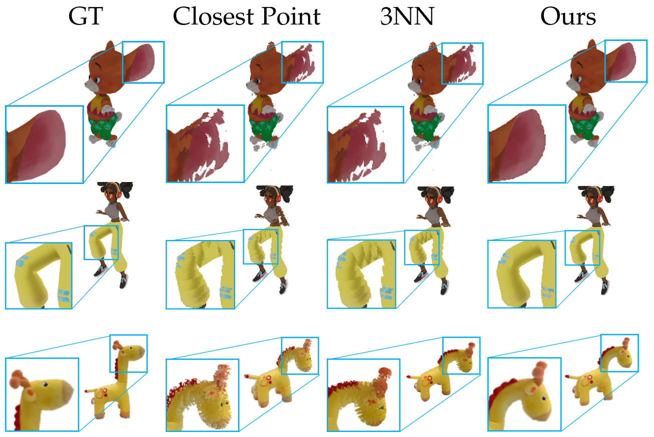


Figure 14. Visual comparisons with the “Closest Point” and “3NN” baselines. The “Closest Point” and “3NN” baselines may cause discontinuities, so the resulting rendering results have obvious artifacts. Note that the last row is a real captured toy giraffe, so ground truth does not exist and we instead visualize the NeRF rendering results before deformation for reference.

the spatial continuity, so the results are more visually satisfactory and have quantitative advantages.

Then we compare our method with mesh rendering baseline on the Lego data from NeRF dataset. It should be noted that although our method uses an explicit triangle mesh representation for interactive editing, it has a certain degree of error tolerance in terms of the mesh reconstruction, and the reconstructed triangle mesh does not need to be perfect. This is because the mesh is only used as an intermediate representation and our final images are still obtained through volume rendering. The direct mesh rendering requires a high quality mesh, and all artifacts on the mesh will appear in the rendered images. As shown in Fig. 15, the reconstructed mesh in the Lego dataset is not of good quality, and the result of mesh rendering is not ideal, while our method can still perform editing, and with the help of volume rendering, desired results can still be obtained.

We also compare the extended version, geometry-aware NeRF-Editing with the original NeRF-Editing [20]. Unlike the original NeRF-Editing which only allows users to manually select control points for editing, geometry-aware NeRF-Editing can automatically generate box abstractions that are convenient for users to edit. At the same time, it can analyze the shape symmetry and maintain the symmetry during

Method	SSIM ↑	LPIPS ↓	PSNR ↑	FID (real) ↓
Closest Point	0.928	0.055	22.38	300.8
3NN	0.941	0.042	24.30	291.7
Ours	**0.975**	**0.024**	**29.62**	**253.7**

Table 1
Quantitative comparison with the "Closest Point" and "3NN" baselines. It can be seen that our framework achieves better results. Note that the first three metrics are calculated on mixamo synthetic data, while the last FID is calculated on real data.

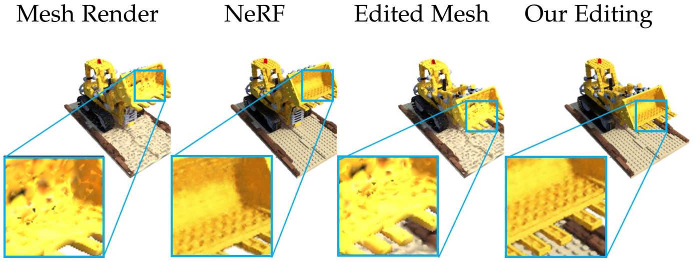


Figure 15. Comparisons with mesh rendering on the synthetic data. Mesh rendering has obvious artifacts when the mesh quality is not good, while this does not affect our editing and image synthesis.

editing, which reduces the editing burden of users and improves the usability of the system. We show the visual comparison results with NeRF-Editing in Fig. 16.

The user is required to deform the same model as close to the same pose as possible using the two different methods. As can be seen from the results, geometry-aware NeRF-Editing can easily maintain the symmetry between the chair legs, and the deformed chair legs are still symmetrical. NeRF-Editing edits the chair legs one by one. To preserve the symmetry of the chair legs, users need to be careful when editing, and the final result is still not symmetrical. Further, we use the symmetry distance error (SDE) [95] to measure the symmetry of the model. Specifically, SDE calculates the average distance from the mirrored copy of the sampled points with respect to the symmetry plane to the mesh model. We calculate the SDE of the reconstructed model before editing and the editing results. The results are shown in Table 2. From the numerical comparison, it can be further seen that our editing results well maintain the symmetry of the original model, while NeRF-Editing increases the symmetry error. In terms of image quality, since our method uses boxes for interaction and cannot automatically generate boxes on the synthetic dataset, maximo, we compare the FID scores with NeRF-Editing on the real scene and the results are shown in Table 3.

We also evaluate the efficiency of our proposed system. The rendering resolution is 480×360. As shown in Table 4, we measure the time of each component, including the preprocessing time, the deformation time, the rendering time, and the total online time. We also show the total time of the original NeRF-Editing for comparison. Note that the total online time does not include preprocessing time, as preprocessing is performed offline and we only consider the time required for online editing. It can be seen that compared with the original version, our extended version has a great improvement in efficiency. Although the preprocessing of our extended version may take some time, the geometric information it brings in will reduce the interactive burden and help maintain the geometric properties of editing results.

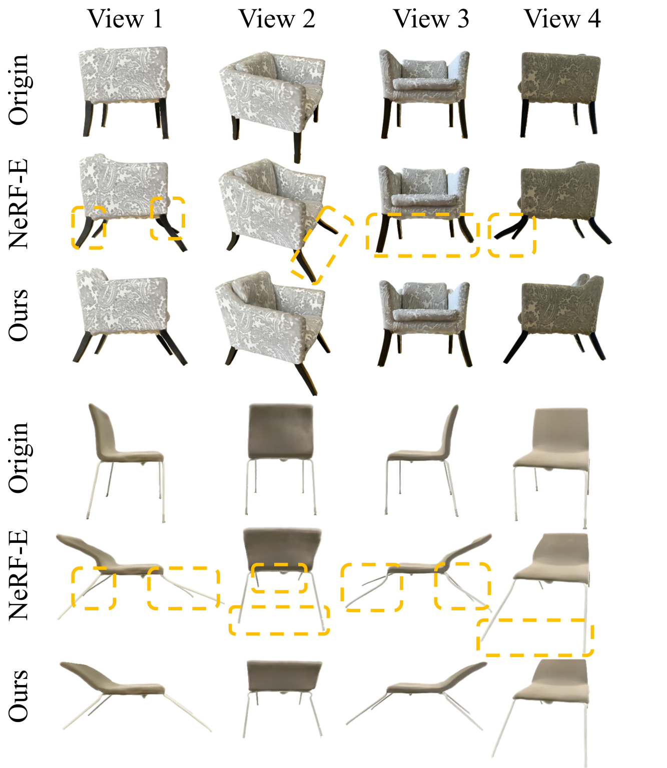


Figure 16. Comparisons with NeRF-Editing [20] (NeRF-E). It can be seen that NeRF-Editing is not easy to maintain symmetry due to editing parts one by one, while our method uses symmetry-constrained editing to ensure symmetry in the editing results.

	Original	NeRF-Editing	w/o sym.	Ours
$SDE(\times 10^{-4})$	1.1596	1.8856	1.9717	**1.1828**

Table 2
Quantitative comparisons with NeRF-Editing [20] and our geometry-aware editing method without symmetry (w/o sym.), along with our full model with symmetry (Ours). We calculate the SDE (symmetry distance error) for the deformed results and the original model. It can be seen that our method well preserves the symmetry of the original model.

4.4 Ablation Study

We conduct ablation studies on the synthetic data with respect to the novel view synthesis results after editing. First, as we introduce a tetrahedral mesh in our method as a proxy between the triangle mesh and the continuous volume, we compare the results of editing on the triangle mesh and editing on the tetrahedral mesh, and verify the necessity of editing on the triangle mesh and transferring deformation by our method. Second, in order to evaluate the influence of the reconstructed triangle mesh on our results, we compare the results of the triangle mesh extracted by the original NeRF and that extracted by NeuS which improves the quality of reconstruction. Tables 5 and 6 summarize the quantitative results of the ablation studies.

Necessity of edit on triangle mesh. Table 5 shows the quantitative comparisons between editing on the tetrahedral mesh and triangle mesh, which indicates that editing on the triangle mesh performs better. The qualitative results are

	NeRF-Editing	w/o seg.	w/o sym.	Box	Ours
FID ↓	254.82	277.96	243.93	290.42	**240.81**

Table 3
Quantitative comparisons with NeRF-Editing [20] and other ablations in terms of image quality. 'Box' represents rendering with the guidance of boxes. We calculate the FID scores that measure the similarity between the image results before and after editing. It can be seen that our method better preserves the distribution of the images.

	Preprocessing	Deformation	Rendering	Total online	Original total
Time	∼ 20min	0.0376s	0.0456s	0.0848s	7.9334s

Table 4
Quantitative comparisons between the original NeRF-Editing [20] and our extended version. We show the preprocessing time, deformation time, rendering time and total online time of our extended version, and total time of the original version for comparison. Note that the total online time does not include preprocessing time. Although preprocessing may take some time, the online editing time of the extended version is much lower than that of the original version.

presented in Fig. 17. The results of editing on the tetrahedral mesh have obvious artifacts due to poor tetrahedral mesh quality.

Impact of mesh quality. Table 6 evaluates the influence of the reconstructed mesh quality to our method. It can be seen that the result of using the mesh from NeuS is better than that of NeRF, but the difference is small. The visual comparisons are shown in Fig. 18, where the results of using the mesh from NeRF have some artifacts in detail, but the overall result is not bad. This illustrates that mesh quality has some but relatively limited effect on our results.

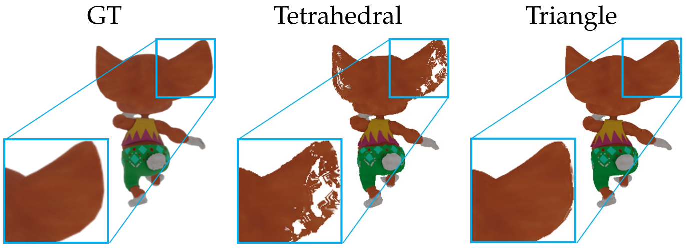


Figure 17. Ablation study of editing on the tetrahedral mesh or triangle mesh. It can be seen that editing on the tetrahedral mesh will bring in artifacts in rendered results.

We conduct three ablation experiments on our extended version, geometry-aware NeRF-Editing. The first two ablation experiments aim at geometry-aware editing. We remove the symmetry information (w/o sym.) or the segmentation information (w/o seg.) during the box editing, and compare with the complete geometry-aware method. Another ablation experiment aims at the rendering approach. Currently, we first use the box handles to drive the deformation of the reconstructed mesh, and then use the mesh to guide the NeRF sampling. An alternative rendering way avoids the introduction of the mesh, which directly uses the box transformation to bend the rays. We compare with this alternative rendering method (box rendering) to illustrate the necessity of introducing the mesh to the editing. Fig. 19 shows the comparison results of the ablation studies.

Editing without segmentation information. The segmentation information can limit the box editing to the corresponding part without affecting other parts. As shown in the second row of Fig. 19, without the segmentation information, the mesh vertices belonging to another part but close to the edge of the parts will also be transformed, resulting in unreasonable results, such as the joint between the seat and the chair leg is elongated together with the chair leg. This is not the case when the segmentation information is introduced, as shown in the last row of Fig. 19.

Method	SSIM ↑	LPIPS ↓	PSNR ↑
Edit on tetrahedral mesh	0.934	0.049	24.37
Edit on triangle mesh	**0.975**	**0.024**	**29.62**

Table 5
Evaluation on the necessity of editing on the triangle mesh. Editing on the triangle mesh leads to better results than directly editing on the tetrahedral mesh.

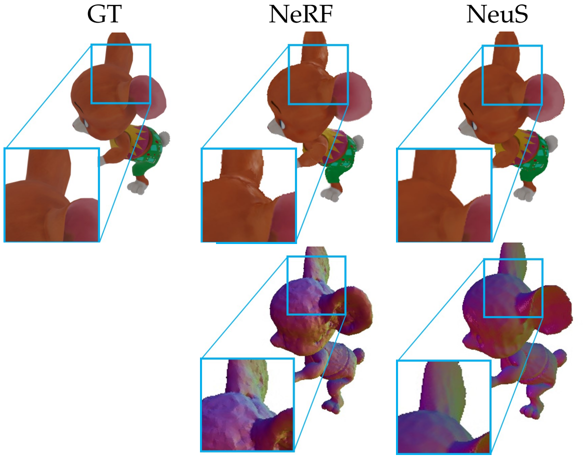


Figure 18. Ablation study of mesh quality. The reconstructed mesh from NeRF is worse than that of NeuS, leading to some artifacts in the rendered results. We visualize the rendered results (first row) and the mesh colored with vertex normals (second row).

Editing without symmetry information. The symmetry information helps to maintain the symmetry of the original model and reduce the editing burden of users, which has been illustrated in the comparison with NeRF-Editing. The third row of Fig. 19 shows the editing results without introducing symmetry. It can be seen that similar to NeRF-Editing, the results do not keep the symmetry of the original model well. We also calculate the SDE of the results, and the results are shown in Table 2. The result of introducing symmetry also has clear advantages quantitatively.

Directly rendering with box transformations. The box transformations can be directly used to bend the rendering rays. Only sampled points within the deformed boxes are preserved. Specifically, each sampled point takes the weighted sum of the transformations of the two nearest boxes as its own transformation. Before the sampled point is fed into the network, its transformation is applied to the coordinates of the sampled point, transforming it back to the original space of NeRF. The results are shown in the fourth row of Fig. 19. It can be seen that the rendering images are incomplete.

4.5 User Study

A user study is conducted to compare the deformation method with symmetry and semantic information with the method without such information. We provide two NeRF models with box handles, semantic and symmetry

Method	SSIM ↑	LPIPS ↓	PSNR ↑
NeRF	0.969	0.027	28.95
NeuS	**0.975**	**0.024**	**29.62**

Table 6
Evaluation on the impact of the extracted mesh quality. The reconstructed mesh from NeuS is better than that from NeRF, leading to better editing results.

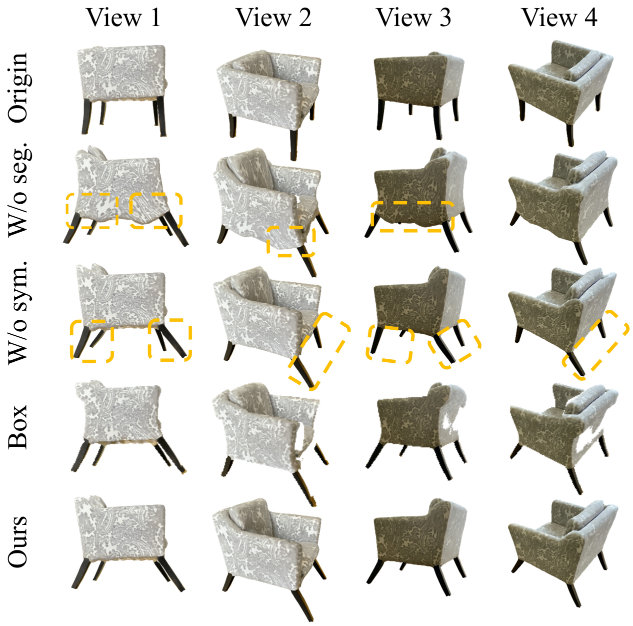


Figure 19. Ablation study. 'Box' represents rendering with the guidance of boxes. Other methods fail to maintain symmetry, or restrict editing within semantic parts, or have issues with rendering, while our method combines geometric information and achieves better results.

information extracted. We then invited 20 participants to try our deformation system, including 6 females and 14 males. 12 are familiar with 3D modeling, while the other 8 are not. Among those 12 individuals, 6 are engaged in research or work in 3D modeling, and 6 are engaged in research or work in other fields of computer graphics. The participants used the deformation method with symmetry and semantic information, denoted as "geometry-aware", and the method without such information, denoted as "non-geometry-aware" to deform the NeRF geometry. They were first asked to use both methods to deform the NeRF geometry into the same target shape as much as possible, then deform to an arbitrary shape. Finally, we asked the participants to score the two methods on six aspects, ease of use, deformation effects, in line with expectations, symmetry preservation, semantic preservation, and interaction burden. Each aspect was scored from 1 to 5, with higher scores indicating better performance on this aspect. We visualize the results in Fig. 20. It can be seen that the geometry-aware deformation method scores higher on all six aspects than the non-geometry-aware deformation method. This shows that the introduction of geometric information in editing has advantages in interactivity, deformation results and preservation of geometric properties.

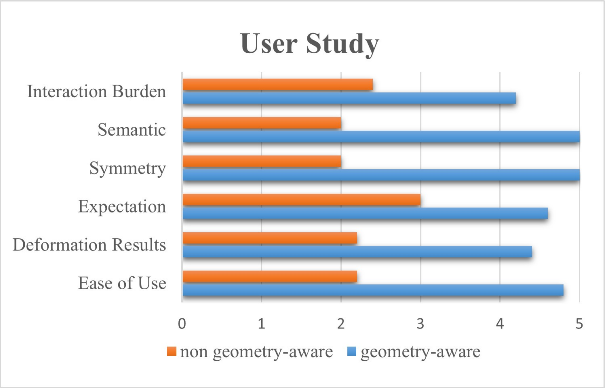


Figure 20. The results of the user study comparing geometry-aware deformation with non-geometry-aware deformation. The participants were asked to score 1 to 5 on six aspects: ease of use, deformation effects, in line with expectations, symmetry preservation, semantic preservation, and interaction burden. Geometry-aware deformation scores higher on all six aspects which demonstrates the benefits of introducing geometric properties.

4.6 Robustness Analysis

In our extended version, we use the method [94] to perform semantic segmentation and box extraction, PRS-Net [95] to analyze the symmetry, and rely on these results in editing. We evaluate the influences of these pre-processing operations on the final editing. To make the reconstructed model suitable for those networks trained on the synthetic data, we scale and align the reconstructed model to match the synthetic models. It should be noted that PRS-Net randomly rotates the model during training, so it is robust to rotation. To further ensure the correctness of symmetry detection, we randomly rotate the model, and then select the symmetry plane with the smallest SDE value as the final result. Wrong rotation and scaling may have an impact on semantic segmentation and box extraction. Because we can always scale the model to the same bounding box size as the ShapeNet data in the end, we only evaluate the influences of wrong rotations. For comparison, we manually rotate the correctly aligned model by a certain degree to deviate from the correct orientation. As shown in Fig. 21 (a), we perform different degrees of deviation. The segmentation and box extraction results are shown in Figs. 21 (b) and (c), respectively. It can be seen that the segmentation results are still ideal, but the box will deviate from the input model, which will be detrimental to our editing operations. To this end, we register the box to the corresponding semantic part. The results after registration are shown in Fig. 21 (d). The registration results fit better to the point cloud and can be used for subsequent editing, reducing the impact of incorrect/inaccurate rotations. Further, we use the registered boxes to deform the corresponding NeRF. We try to keep the edits the same in both cases. Fig. 22 compares the editing results when the alignment is accurate and the editing results when the alignment is inaccurate. It can be seen that after correcting the deviation of the generated boxes, the editing when the alignment is deviated can still ensure the ideal results.

4.7 Limitations

Our method is the first step for geometric shape deformation on NeRFs and still has several limitations. First

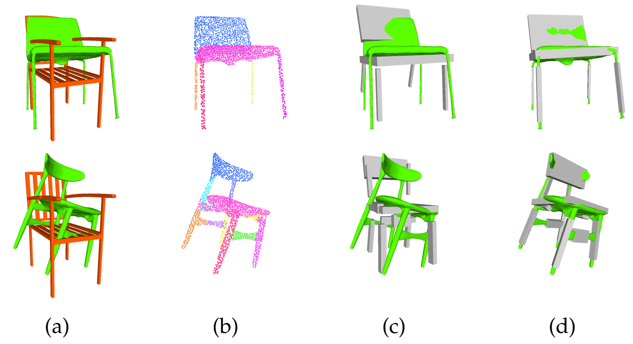


Figure 21. Evaluations of the influences of wrong rotations to segmentation and box abstraction. (a) Models with wrong rotations, (b) Segmentation results, (c) box abstraction results, (d) Refined box results.

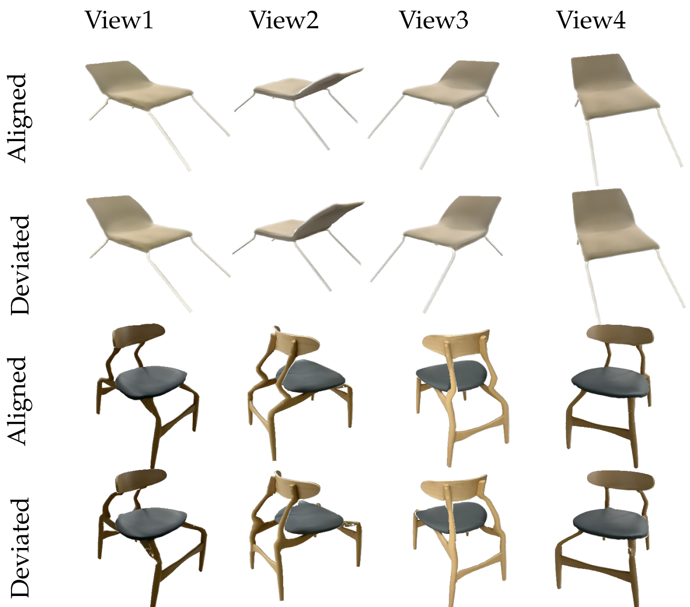


Figure 22. Results of wrong rotations (row "Deviated") compared with correct rotation (row "Aligned"). Different columns show different views.

of all, the biggest limitation is that we cannot modify the color as well as the lighting and shadow based on the editing results. If an object part that is in the shadow during capturing is deformed to face the light, its color will still be dim instead of bright, as shown in Fig. 23. Similarly, the shadow of the respective parts of the combined objects are only affected by the original environment. The newly added part will not change the shadow of other parts, and the combination of parts in different environments may cause inconsistency, as shown in Fig. 24. This could be dealt with by incorporating some recent NeRF-based relighting work [6], [55] to achieve correct color rendering by decoupling lighting. Second, limited by the geometry analysis method we use, we can only automatically generate cube handles and extract segmentation information and symmetry information for specific categories of objects. For example, as [94] is only trained on chair, table, animal and airplane categories, the editing objects we show in this paper are also mainly chair, table and animal models. But this does not limit the implementation of our interactive editing method, and as more data becomes available, the method [94] can also be trained on other categories. Users can also draw the cubes by themselves to edit. Last, although we have verified that inaccurate alignment has little impact on geometry-aware editing, wrong geometric analysis results can have an impact on editing. Should this happen, it is better to edit without using segmentation information and symmetry information to assist editing.

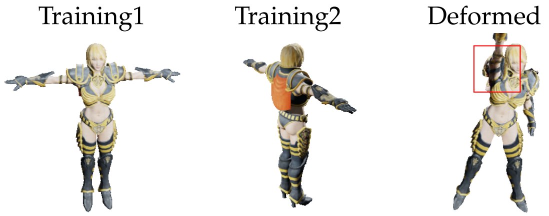


Figure 23. A failure case for deformation. Our approach does not edit the appearance along with geometry deformation. In this example, since the underarm is occluded from the light in the T-pose during training, it will always be gloomy even when the woman raises her arm, which is implausible.

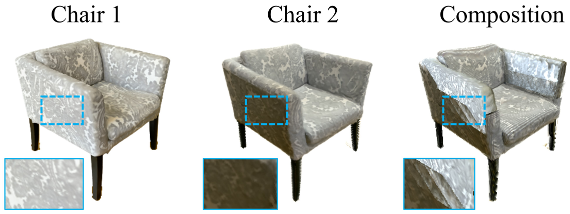


Figure 24. A failure case for composition. Our method cannot eliminate or change the lighting and shadow, so when combining parts from significantly different lighting environments, lighting and shadow inconsistencies will appear.

5 Conclusion

In this paper, we propose the first method to support user-controlled shape deformation to the geometry of neural radiance field network. By establishing a correspondence between the explicit mesh representation and the implicit volume representation, our method can use the well-developed triangular mesh deformation method to deform the implicit representation. With the novel view synthesis capability of NeRF, users can visualize the editing results from arbitrary views. Our method is suitable for general real scenes which can edit scene objects including human bodies, animals, man-made models, etc. Compared with the previous editing methods for NeRF, our method has a higher degree of freedom and can support the editing of details. Aiming to reduce the interaction burden of the user, we propose an extended version to support interactive geometry-aware shape deformation to the geometry of NeRF. With the synthetic model's data priors, we extract cube abstractions for the real reconstructed models as handles for user editing. The segmentation and symmetry information extracted simultaneously makes editing more efficient and maintains the geometric properties of the model. We build an interactive editing system based on this method, where users can use the cube handles to deform the geometry of NeRF and synthesize novel view results in real-time. Further, using segmentation information, our method can also supports semantic part-level NeRF compositions. Compared with the original version, the extended version has advantages in efficiency and maintaining the geometric properties of the model. In the future, we will explore how to directly perform geometric analysis of models modeled by NeRF, as well as incorporating relighting methods to adjust lighting and shadow during editing.

Acknowledgment

This work was supported by grants from the National Natural Science Foundation of China (No. 62061136007), the Beijing Municipal Natural Science Foundation for Distinguished Young Scholars (No. JQ21013), the Royal Society Newton Advanced Fellowship (No. NAF\R2\192151), and the Youth Innovation Promotion Association CAS.

References

[1] B. Mildenhall, P. P. Srinivasan, M. Tancik, J. T. Barron, R. Ramamoorthi, and R. Ng, "NeRF: Representing scenes as neural radiance fields for view synthesis," in *European Conference on Computer Vision*. Springer, 2020, pp. 405–421.

[2] L. Liu, J. Gu, K. Zaw Lin, T.-S. Chua, and C. Theobalt, "Neural sparse voxel fields," *Advances in Neural Information Processing Systems*, vol. 33, 2020.

[3] A. Yu, R. Li, M. Tancik, H. Li, R. Ng, and A. Kanazawa, "Plenoctrees for real-time rendering of neural radiance fields," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 5752–5761.

[4] S. J. Garbin, M. Kowalski, M. Johnson, J. Shotton, and J. Valentin, "FastNeRF: High-fidelity neural rendering at 200fps," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 14 346–14 355.

[5] A. Yu, V. Ye, M. Tancik, and A. Kanazawa, "pixelNeRF: Neural radiance fields from one or few images," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 4578–4587.

[6] M. Boss, R. Braun, V. Jampani, J. T. Barron, C. Liu, and H. Lensch, "NeRD: Neural reflectance decomposition from image collections," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 12 684–12 694.

[7] P. P. Srinivasan, B. Deng, X. Zhang, M. Tancik, B. Mildenhall, and J. T. Barron, "NeRV: Neural reflectance and visibility fields for relighting and view synthesis," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 7495–7504.

[8] A. Pumarola, E. Corona, G. Pons-Moll, and F. Moreno-Noguer, "D-NeRF: Neural radiance fields for dynamic scenes," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 10 318–10 327.

[9] A. Jain, M. Tancik, and P. Abbeel, "Putting NeRF on a diet: Semantically consistent few-shot view synthesis," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 5885–5894.

[10] Y.-J. Yuan, Y.-K. Lai, Y.-H. Huang, L. Kobbelt, and L. Gao, "Neural radiance fields from sparse rgb-d images for high-quality view synthesis," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2022.

[11] Y.-J. Yuan, Y.-K. Lai, T. Wu, L. Gao, and L. Liu, "A revisit of shape editing techniques: From the geometric to the neural viewpoint," *Journal of Computer Science and Technology*, vol. 36, no. 3, pp. 520–554, 2021.

[12] J. L. Schönberger, E. Zheng, M. Pollefeys, and J.-M. Frahm, "Pixelwise view selection for unstructured multi-view stereo," in *European Conference on Computer Vision*, 2016.

[13] S. Liu, X. Zhang, Z. Zhang, R. Zhang, J.-Y. Zhu, and B. Russell, "Editing conditional radiance fields," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 5773–5783.

[14] J. Zhang, X. Liu, X. Ye, F. Zhao, Y. Zhang, M. Wu, Y. Zhang, L. Xu, and J. Yu, "Editable free-viewpoint video using a layered neural representation," *ACM Transactions on Graphics (TOG)*, vol. 40, no. 4, pp. 1–18, 2021.

[15] B. Yang, Y. Zhang, Y. Xu, Y. Li, H. Zhou, H. Bao, G. Zhang, and Z. Cui, "Learning object-compositional neural radiance field for editable scene rendering," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, October 2021, pp. 13 779–13 788.

[16] V. Lazova, V. Guzov, K. Olszewski, S. Tulyakov, and G. Pons-Moll, "Control-nerf: Editable feature volumes for scene rendering and manipulation," *arXiv preprint arXiv:2204.10850*, 2022.

[17] K. Kania, K. M. Yi, M. Kowalski, T. Trzciński, and A. Tagliasacchi, "Conerf: Controllable neural radiance fields," *arXiv preprint arXiv:2112.01983*, 2021.

[18] E. Tretschk, A. Tewari, V. Golyanik, M. Zollhofer, C. Lassner, and C. Theobalt, "Non-rigid neural radiance fields: Reconstruction and novel view synthesis of a dynamic scene from monocular video," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 12 959–12 970.

[19] S. Peng, J. Dong, Q. Wang, S. Zhang, Q. Shuai, X. Zhou, and H. Bao, "Animatable neural radiance fields for modeling dynamic human bodies," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 14 314–14 323.

[20] Y.-J. Yuan, Y.-T. Sun, Y.-K. Lai, Y. Ma, R. Jia, and L. Gao, "Nerf-editing: Geometry editing of neural radiance fields," in *Computer Vision and Pattern Recognition (CVPR)*, 2022.

[21] G. Chaurasia, S. Duchêne, O. Sorkine-Hornung, and G. Drettakis, "Depth synthesis and local warps for plausible image-based navigation," *ACM Transactions on Graphics (TOG)*, vol. 32, pp. 30:1–30:12, 2013.

[22] P. Hedman, S. Alsisan, R. Szeliski, and J. Kopf, "Casual 3D photography," *ACM Transactions on Graphics (TOG)*, vol. 36, pp. 1 – 15, 2017.

[23] N. Snavely, S. Seitz, and R. Szeliski, "Photo tourism: exploring photo collections in 3D," in *SIGGRAPH 2006*, 2006.

[24] P. Hedman, T. Ritschel, G. Drettakis, and G. Brostow, "Scalable inside-out image-based rendering," *ACM Transactions on Graphics (TOG)*, vol. 35, pp. 1 – 11, 2016.

[25] S. Gortler, R. Grzeszczuk, R. Szeliski, and M. Cohen, "The lumigraph," *Proceedings of the 23rd annual conference on Computer Graphics and Interactive Techniques*, 1996.

[26] M. Levoy and P. Hanrahan, "Light field rendering," *Proceedings of the 23rd Annual Conference on Computer Graphics and interactive techniques*, 1996.

[27] R. Szeliski and P. Golland, "Stereo matching with transparency and matting," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*. IEEE, 1998, pp. 517–524.

[28] A. Tewari, O. Fried, J. Thies, V. Sitzmann, S. Lombardi, K. Sunkavalli, R. Martin-Brualla, T. Simon, J. M. Saragih, M. Nießner, R. Pandey, S. Fanello, G. Wetzstein, J.-Y. Zhu, C. Theobalt, M. Agrawala, E. Shechtman, D. B. Goldman, and M. Zollhofer, "State of the art on neural rendering," *Computer Graphics Forum*, vol. 39, 2020.

[29] A. Tewari, O. Fried, J. Thies, V. Sitzmann, S. Lombardi, Z. Xu, T. Simon, M. Nießner, E. Tretschk, L. Liu, B. Mildenhall, P. Srinivasan, R. Pandey, S. Orts-Escolano, S. Fanello, M. Guo, G. Wetzstein, J. y Zhu, C. Theobalt, M. Agrawala, D. B. Goldman, and M. Zollhöfer, "Advances in neural rendering," *ACM SIGGRAPH 2021 Courses*, 2021.

[30] V. Sitzmann, J. Thies, F. Heide, M. Nießner, G. Wetzstein, and M. Zollhofer, "DeepVoxels: Learning persistent 3D feature embeddings," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 2437–2446.

[31] S. Lombardi, T. Simon, J. Saragih, G. Schwartz, A. Lehrmann, and Y. Sheikh, "Neural volumes: learning dynamic renderable volumes from images," *ACM Transactions on Graphics (TOG)*, vol. 38, no. 4, pp. 1–14, 2019.

[32] K.-A. Aliev, A. Sevastopolsky, M. Kolos, D. Ulyanov, and V. Lempitsky, "Neural point-based graphics," in *European Conference on Computer Vision*. Springer, 2020, pp. 696–712.

[33] P. Dai, Y. Zhang, Z. Li, S. Liu, and B. Zeng, "Neural point cloud rendering via multi-plane projection," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 7830–7839.

[34] A. Chen, M. Wu, Y. Zhang, N. Li, J. Lu, S. Gao, and J. Yu, "Deep surface light fields," in *Proceedings of the ACM on Computer Graphics and Interactive Techniques*, vol. 1, no. 1. ACM New York, NY, USA, 2018, pp. 1–17.

[35] J. Thies, M. Zollhöfer, and M. Nießner, "Deferred neural rendering: Image synthesis using neural textures," *ACM Transactions on Graphics (TOG)*, vol. 38, no. 4, pp. 1–12, 2019.

[36] G. Riegler and V. Koltun, "Free view synthesis," in *European Conference on Computer Vision*. Springer, 2020, pp. 623–640.

[37] ——, "Stable view synthesis," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2021.

[38] T. Zhou, R. Tucker, J. Flynn, G. Fyffe, and N. Snavely, "Stereo magnification: learning view synthesis using multiplane images," *ACM Transactions on Graphics (TOG)*, vol. 37, no. 4, pp. 1–12, 2018.

[39] B. Mildenhall, P. P. Srinivasan, R. Ortiz-Cayon, N. K. Kalantari, R. Ramamoorthi, R. Ng, and A. Kar, "Local light field fusion: Practical view synthesis with prescriptive sampling guidelines," *ACM Transactions on Graphics (TOG)*, vol. 38, no. 4, pp. 1–14, 2019.

[40] Z. Li, W. Xian, A. Davis, and N. Snavely, "Crowdsampling the plenoptic function," in *European Conference on Computer Vision*. Springer, 2020, pp. 178–196.

[41] V. Sitzmann, M. Zollhöfer, and G. Wetzstein, "Scene representation networks: continuous 3D-structure-aware neural scene representations," in *Proceedings of the 33rd International Conference on Neural Information Processing Systems*, 2019, pp. 1121–1132.

[42] P. Kellnhofer, L. C. Jebe, A. Jones, R. Spicer, K. Pulli, and G. Wetzstein, "Neural lumigraph rendering," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 4287–4297.

[43] S. Wizadwongsa, P. Phongthawee, J. Yenphraphai, and S. Suwajanakorn, "NeX: Real-time view synthesis with neural basis expansion," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 8534–8543.

[44] K. Zhang, G. Riegler, N. Snavely, and V. Koltun, "NeRF++: Analyzing and improving neural radiance fields," *ArXiv*, vol. abs/2010.07492, 2020.

[45] S. Peng, Y. Zhang, Y. Xu, Q. Wang, Q. Shuai, H. Bao, and X. Zhou, "Neural body: Implicit neural representations with structured latent codes for novel view synthesis of dynamic humans," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 9054–9063.

[46] G. Gafni, J. Thies, M. Zollhofer, and M. Nießner, "Dynamic neural radiance fields for monocular 4D facial avatar reconstruction," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 8649–8658.

[47] Z. Li, S. Niklaus, N. Snavely, and O. Wang, "Neural scene flow fields for space-time view synthesis of dynamic scenes," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 6498–6508.

[48] K. Park, U. Sinha, J. T. Barron, S. Bouaziz, D. B. Goldman, S. M. Seitz, and R. Martin-Brualla, "Nerfies: Deformable neural radiance fields," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 5865–5874.

[49] K. Park, U. Sinha, P. Hedman, J. T. Barron, S. Bouaziz, D. B. Goldman, R. Martin-Brualla, and S. M. Seitz, "HyperNeRF: a higher-dimensional representation for topologically varying neural radiance fields," *ACM Transactions on Graphics (TOG)*, vol. 40, no. 6, pp. 1–12, 2021.

[50] T. Müller, A. Evans, C. Schied, and A. Keller, "Instant neural graphics primitives with a multiresolution hash encoding," *ACM Trans. Graph.*, vol. 41, no. 4, pp. 102:1–102:15, Jul. 2022. [Online]. Available: https://doi.org/10.1145/3528223.3530127

[51] P. Hedman, P. P. Srinivasan, B. Mildenhall, J. T. Barron, and P. Debevec, "Baking neural radiance fields for real-time view synthesis," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 5875–5884.

[52] C. Reiser, S. Peng, Y. Liao, and A. Geiger, "KiloNeRF: Speeding up neural radiance fields with thousands of tiny MLPs," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 14 335–14 345.

[53] Q. Wang, Z. Wang, K. Genova, P. Srinivasan, H. Zhou, J. T. Barron, R. Martin-Brualla, N. Snavely, and T. Funkhouser, "IBRNet: Learning multi-view image-based rendering," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 4690–4699.

[54] A. Chen, Z. Xu, F. Zhao, X. Zhang, F. Xiang, J. Yu, and H. Su, "MVSNeRF: Fast generalizable radiance field reconstruction from multi-view stereo," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 14 124–14 133.

[55] X. Zhang, P. P. Srinivasan, B. Deng, P. Debevec, W. T. Freeman, and J. T. Barron, "NeRFactor: Neural factorization of shape and reflectance under an unknown illumination," *ACM Transactions on Graphics (TOG)*, vol. 40, no. 6, pp. 1–18, 2021.

[56] K. Jiang, S.-Y. Chen, H. Fu, and L. Gao, "Nerffacelighting: Implicit and disentangled face lighting representation leveraging generative prior in neural radiance fields," *ACM Transactions on Graphics*, vol. 42, no. 3, pp. 1–18, 2023.

[57] Y.-H. Huang, Y. He, Y.-J. Yuan, Y.-K. Lai, and L. Gao, "Stylizednerf: Consistent 3d scene stylization as stylized nerf via 2d-3d mutual learning," in *Computer Vision and Pattern Recognition (CVPR)*, 2022.

[58] K. Jiang, S.-Y. Chen, F.-L. Liu, H. Fu, and L. Gao, "Nerffaceediting: Disentangled face editing in neural radiance fields," in *SIGGRAPH Asia 2022 Conference Papers*, 2022, pp. 1–9.

[59] T. Wu, J.-M. Sun, Y.-K. Lai, and L. Gao, "De-nerf: Decoupled neural radiance fields for view-consistent appearance editing and high-frequency environmental relighting," in *SIGGRAPH 2023 Conference Papers*, 2023.

[60] Y.-H. Huang, Y.-P. Cao, Y.-K. Lai, Y. Shan, and L. Gao, "Nerf-texture: Texture synthesis with neural radiance fields," in *SIGGRAPH 2023 Conference Papers*, 2023.

[61] L. Gao, F.-L. Liu, S.-Y. Chen, K. Jiang, C. Li, Y.-K. Lai, and H. Fu, "SketchFaceNeRF: Sketch-based facial generation and editing in neural radiance fields," *ACM Transactions on Graphics*, 2023.

[62] Chong Bao and Bangbang Yang, Z. Junyi, B. Hujun, Z. Yinda, C. Zhaopeng, and Z. Guofeng, "Neumesh: Learning disentangled neural mesh-based implicit field for geometry and texture editing," in *European Conference on Computer Vision (ECCV)*, 2022.

[63] T. Xu and T. Harada, "Deforming radiance fields with cages," in *ECCV*, 2022.

[64] L. Gao, G. Zhang, and Y. Lai, "L p shape deformation," *Science China Information Sciences*, vol. 55, no. 5, pp. 983–993, 2012.

[65] S.-Y. Chen, L. Gao, Y.-K. Lai, and S. Xia, "Rigidity controllable as-rigid-as-possible shape deformation," *Graphical Models*, vol. 91, pp. 13–21, 2017.

[66] O. Sorkine, D. Cohen-Or, Y. Lipman, M. Alexa, C. Rössl, and H.-P. Seidel, "Laplacian surface editing," in *Proceedings of the 2004 Eurographics/ACM SIGGRAPH Symposium on Geometry Processing*, 2004, pp. 175–184.

[67] O. Sorkine, "Laplacian mesh processing," *Eurographics (STARs)*, vol. 29, 2005.

[68] Y. Lipman, O. Sorkine, M. Alexa, D. Cohen-Or, D. Levin, C. Rössl, and H.-P. Seidel, "Laplacian framework for interactive mesh editing," *International Journal of Shape Modeling*, vol. 11, no. 01, pp. 43–61, 2005.

[69] Y. Yu, K. Zhou, D. Xu, X. Shi, H. Bao, B. Guo, and H.-Y. Shum, "Mesh editing with Poisson-based gradient field manipulation," in *ACM SIGGRAPH 2004 Papers*, 2004, pp. 644–651.

[70] O.-C. Au, C.-L. Tai, L. Liu, and H. Fu, "Dual Laplacian editing for meshes," *IEEE Transactions on Visualization and Computer Graphics*, vol. 12, no. 3, pp. 386–395, 2006.

[71] O. Sorkine-Hornung and M. Alexa, "As-rigid-as-possible surface modeling," in *Symposium on Geometry Processing*, 2007.

[72] N. Magnenat-Thalmann, R. Laperrire, and D. Thalmann, "Joint-dependent local deformations for hand animation and object grasping," in *In Proceedings on Graphics interface'88*. Citeseer, 1988.

[73] A. Jacobson, I. Baran, L. Kavan, J. Popović, and O. Sorkine, "Fast automatic skinning transformations," *ACM Transactions on Graphics (TOG)*, vol. 31, no. 4, pp. 1–10, 2012.

[74] T. W. Sederberg and S. R. Parry, "Free-form deformation of solid geometric models," *SIGGRAPH Comput. Graph.*, vol. 20, no. 4, p. 151–160, 1986.

[75] Y. Zhang, J. Zheng, and Y. Cai, "Proxy-driven free-form deformation by topology-adjustable control lattice," *Computers & Graphics*, vol. 89, pp. 167–177, 2020.

[76] W. Yifan, N. Aigerman, V. G. Kim, S. Chaudhuri, and O. Sorkine-Hornung, "Neural cages for detail-preserving 3D deformations," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 75–83.

[77] A. Jacobson, I. Baran, J. Popovic, and O. Sorkine, "Bounded biharmonic weights for real-time deformation," *ACM Transactions on Graphics (TOG)*, vol. 30, no. 4, pp. 78–1, 2011.

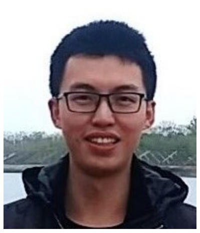
[78] Y.-J. Yuan, Y.-K. Lai, T. Wu, S. Xia, and L. Gao, "Data-driven weight optimization for real-time mesh deformation," *Graphical Models*, vol. 104, p. 101037, 2019.

[79] M. S. Floater, "Mean value coordinates," *Computer Aided Geometric Design*, vol. 20, no. 1, pp. 19–27, 2003.

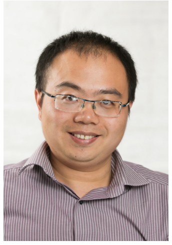
[80] F. Bogo, J. Romero, M. Loper, and M. J. Black, "FAUST: Dataset and evaluation for 3D mesh registration," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. IEEE, 2014.

[81] R. W. Sumner, M. Zwicker, C. Gotsman, and J. Popović, “Mesh-based inverse kinematics,” *ACM Transactions on Graphics (TOG)*, vol. 24, no. 3, pp. 488–495, 2005.
[82] L. Gao, Y.-K. Lai, D. Liang, S.-Y. Chen, and S. Xia, “Efficient and flexible deformation representation for data-driven surface modeling,” *ACM Transactions on Graphics (TOG)*, vol. 35, no. 5, pp. 158:1–158:17, 2016.
[83] L. Gao, Y.-K. Lai, J. Yang, Z. Ling-Xiao, S. Xia, and L. Kobbelt, “Sparse data driven mesh deformation,” *IEEE Transactions on Visualization and Computer Graphics*, vol. 27, no. 3, pp. 2085–2100, 2019.
[84] Q. Tan, L. Gao, Y.-K. Lai, and S. Xia, “Variational autoencoders for deforming 3D mesh models,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2018, pp. 5841–5850.
[85] J. Yang, L. Gao, Q. Tan, Y.-H. Huang, S. Xia, and Y.-K. Lai, “Multi-scale mesh deformation component analysis with attention-based autoencoders,” *IEEE Transactions on Visualization and Computer Graphics*, 2021.
[86] L. Liu, Y. Zheng, D. Tang, Y. Yuan, C. Fan, and K. Zhou, “NeuroSkinning: Automatic skin binding for production characters with deep graph networks,” *ACM Transactions on Graphics (TOG)*, vol. 38, no. 4, pp. 1–12, 2019.
[87] Z. Xu, Y. Zhou, E. Kalogerakis, C. Landreth, and K. Singh, “RigNet: Neural rigging for articulated characters,” *ACM Transactions on Graphics (TOG)*, vol. 39, no. 4, pp. 1–14, 2020.
[88] Y. Deng, J. Yang, and X. Tong, “Deformed implicit field: Modeling 3D shapes with learned dense correspondence,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021.
[89] J. T. Kajiya and B. P. Von Herzen, “Ray tracing volume densities,” *ACM SIGGRAPH Computer Graphics*, vol. 18, no. 3, pp. 165–174, 1984.
[90] W. E. Lorensen and H. E. Cline, “Marching cubes: A high resolution 3D surface construction algorithm,” *ACM Siggraph Computer Graphics*, vol. 21, no. 4, pp. 163–169, 1987.
[91] P. Wang, L. Liu, Y. Liu, C. Theobalt, T. Komura, and W. Wang, “NeuS: Learning neural implicit surfaces by volume rendering for multi-view reconstruction,” in *Advances in Neural Information Processing Systems*, vol. 34, 2021.
[92] Y. Hu, T. Schneider, B. Wang, D. Zorin, and D. Panozzo, “Fast tetrahedral meshing in the wild,” *ACM Transactions on Graphics (TOG)*, vol. 39, pp. 117:1 – 117:18, 2020.
[93] P. J. Besl and N. D. McKay, “Method for registration of 3-D shapes,” in *Sensor fusion IV: control paradigms and data structures*, vol. 1611. Spie, 1992, pp. 586–606.
[94] K. Yang and X. Chen, “Unsupervised learning for cuboid shape abstraction via joint segmentation from point clouds,” *ACM Transactions on Graphics (TOG)*, vol. 40, no. 4, pp. 1–11, 2021.
[95] L. Gao, L.-X. Zhang, H.-Y. Meng, Y.-H. Ren, Y.-K. Lai, and L. Kobbelt, “PRS-Net: Planar reflective symmetry detection net for 3D models,” *IEEE Transactions on Visualization and Computer Graphics*, vol. 27, no. 6, pp. 3007–3018, 2020.
[96] M. Alexa, “Linear combination of transformations,” *ACM Transactions on Graphics (TOG)*, vol. 21, no. 3, pp. 380–387, 2002.
[97] A. Inc, “Mixamo,” https://www.mixamo.com.
[98] Z. Wang, A. C. Bovik, H. R. Sheikh, E. P. Simoncelli *et al.*, “Image quality assessment: from error visibility to structural similarity,” *IEEE Transactions on Image Processing*, vol. 13, no. 4, pp. 600–612, 2004.
[99] R. Zhang, P. Isola, A. A. Efros, E. Shechtman, and O. Wang, “The unreasonable effectiveness of deep features as a perceptual metric,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2018, pp. 586–595.
[100] M. Heusel, H. Ramsauer, T. Unterthiner, B. Nessler, and S. Hochreiter, “GANs trained by a two time-scale update rule converge to a local nash equilibrium,” in *Advances in Neural Information Processing Systems*, vol. 30, 2017.
[101] R. W. Sumner and J. Popović, “Deformation transfer for triangle meshes,” *ACM Transactions on Graphics (TOG)*, vol. 23, no. 3, pp. 399–405, 2004.

Yu-Jie Yuan received the bachelor’s degree in mathematics from Xi’an Jiaotong University in 2018. He is currently a Ph.D. candidate in the Institute of Computing Technology, Chinese Academy of Sciences. His research interests include computer graphics and neural rendering.



Yang-Tian Sun received the Master’s degree from the Institute of Computing Technology, Chinese Academy of Sciences in 2022. His research interests include computer graphics and neural rendering.



Yu-Kun Lai received his bachelor’s degree and PhD degree in computer science from Tsinghua University in 2003 and 2008, respectively. He is currently a Professor in the School of Computer Science & Informatics, Cardiff University. His research interests include computer graphics, geometry processing, image processing and computer vision. He is on the editorial boards of *IEEE Transactions on Visualization and Computer Graphics* and *The Visual Computer*.

Yuewen Ma received his PhD degree from Nanyang Technological University, Singapore in 2013. He has been engaged in the research and product of computer graphics and 3D vision for a long time. He is currently the leader of 3D reconstruction at ByteDance Pico.

Rongfei Jia received his PhD degree from Beihang University. He focuses on 3D deep learning research and has realized automatic 3D reconstruction of commodities, automatic layout generation of real houses, and intelligent matching generation. He has published many papers at conferences such as NeurIPS, ICCV, and KDD. He also won the first Chinagraph open-source dataset award.

Leif Kobbelt is a full professor and the head of the Computer Graphics Group at the RWTH Aachen University, Germany. His research interests include all areas of Computer Graphics and Geometry Processing with a focus on multi resolution and free-form modeling as well as the efficient handling of polygonal mesh data. He received his master’s degree in 1992 and Ph.D. in 1994 from the University of Karlsruhe, Germany.

Lin Gao received the bachelor's degree in mathematics from Sichuan University and the PhD degree in computer science from Tsinghua University. He is currently an Associate Professor at the Institute of Computing Technology, Chinese Academy of Sciences. He has been awarded the Newton Advanced Fellowship from the Royal Society and the Asia Graphics Association young researcher award. His research interests include computer graphics and geometric processing.