NeRF-Editing: Geometry Editing of Neural Radiance Fields
Supplementary Material

Yu-Jie Yuan [1,2,†] **Yang-Tian Sun** [1,2,†] **Yu-Kun Lai** [3]
Yuewen Ma [4] **Rongfei Jia** [4] **Lin Gao** [1,2*]
[1]Beijing Key Laboratory of Mobile Computing and Pervasive Device,
Institute of Computing Technology, Chinese Academy of Sciences
[2]School of Computer and Control Engineering, University of Chinese Academy of Sciences
[3]School of Computer Science & Informatics, Cardiff University [4]Alibaba Group
{yuanyujie, sunyangtian, gaolin}@ict.ac.cn LaiY4@cardiff.ac.uk
{yuewen.my, rongfei.jrf}@alibaba-inc.com

1. Overview

This supplementary document provides the implementation details and more results that accompany the paper. It contains four parts, including implementation details, intermediate results, more editing results, and an example result of deformation transfer.

- Section 2 provides some key derivations of the algorithm, the details of self-captured datasets and training details.
- Section 3 provides more editing results on both synthetic and real captured datasets.
- Section 4 provides the intermediate results in the editing process, including visualization of the triangular meshes and tetrahedral meshes.
- Section 5 provides an example result of the deformation transfer between a video clip and a head sculpture.

2. Implementation Details

In this section, we will go over some implementation details, including the derivation of ARAP (as-rigid-as-possible) deformation of a tetrahedral mesh under the constraints of a deformed triangular mesh, calculation of barycentric coordinates for sampled points, self-captured dataset details and training details.

2.1. Derivation of As-Rigid-As-Possible (ARAP) Deformation of Tetrahedral Mesh

Firstly, we explain how a tetrahedral mesh is deformed under the constraints of a triangular mesh. Following the main paper, the deformed triangular mesh is denoted as S' with vertex positions $\boldsymbol{v}'_i$. The tetrahedral mesh before and after deformation are denoted as T and T' with vertex positions $\boldsymbol{t}_k$ and $\boldsymbol{t}'_k$ respectively. The deformation of the tetrahedral mesh is formulated as follows:

$$\begin{aligned} &\min E(T') = \sum_{k=1}^{m} \tilde{w}_i \sum_{j\in N(k)} w_{kj} \|(\boldsymbol{t}'_k - \boldsymbol{t}'_j) - \mathbf{R}_k(\boldsymbol{t}_k - \boldsymbol{t}_j)\|^2, \\ &\text{subject to } \boldsymbol{A}\boldsymbol{t}' = \boldsymbol{v}', \end{aligned} \tag{1}$$

where $\mathbf{R}_k$ is the local rotation at vertex k, $N(k)$ is the set of adjacent vertices of vertex k, m is the total number of tetrahedral vertices, w_{kj} is the cotangent weight, $\boldsymbol{A}$ is the barycentric weight matrix. $\tilde{w}_i$ is the cell weight which is set to 1 following [6]. This quadratic optimization problem with linear constraints can be converted into linear equations using the Lagrangian multiplier method. By introducing several new variables λ_i, the constrained optimization problem in Eq. 1 can be converted into an unconstrained minimization problem:

$$\begin{aligned} \min G(T') = &\sum_{k=1}^{m} \sum_{j\in N(k)} w_{kj} \|(\boldsymbol{t}'_k - \boldsymbol{t}'_j) - \mathbf{R}_k(\boldsymbol{t}_k - \boldsymbol{t}_j)\|^2, \\ &+ \sum_{i=1}^{n} \lambda_i (\boldsymbol{A}_i \boldsymbol{t}' - \boldsymbol{v}'_i), \end{aligned} \tag{2}$$

where $\boldsymbol{A}_i$ is the i-th row of the matrix $\boldsymbol{A}$, and $\boldsymbol{v}_i$ is the same as defined before. The minimization problem can be efficiently solved by alternately optimizing local rotations $\mathbf{R}_k$ and deformed positions $\boldsymbol{t}'_k$. The solution of local rotations $\mathbf{R}_k$ is the same as the original ARAP [6]. Here, we focus on how to solve the deformed positions $\boldsymbol{t}'_k$ supposing that

†: Authors contributed equally
*Corresponding Author is Lin Gao (gaolin@ict.ac.cn)

the local rotations $\mathbf{R}_k$ is known. We consider the partial derivative of $G(T')$ w.r.t. $\boldsymbol{t}'_k$:

$$\frac{\partial G(T')}{\partial \boldsymbol{t}'_k} = \sum_{j \in N(k)} 4w_{kj}((\boldsymbol{t}'_k - \boldsymbol{t}'_j) - \frac{1}{2}(\mathbf{R}_k + \mathbf{R}_j)(\boldsymbol{t}_k - \boldsymbol{t}_j)) + \sum_{l=1}^{L} \lambda_l a_l, \tag{3}$$

where a_l is the non-zero coefficient corresponding to $\boldsymbol{t}_k$ in $\boldsymbol{A}$, and L is the total number of the non-zero coefficients corresponding to $\boldsymbol{t}_k$. And then we consider $\frac{\partial G(T')}{\partial \boldsymbol{t}'_k} = 0$, which can lead us to the following sparse linear system of equations:

$$\sum_{j \in N(k)} w_{kj}(\boldsymbol{t}'_k - \boldsymbol{t}'_j) + \sum_{l=1}^{L} \frac{\lambda_l a_l}{4} = \sum_{j \in N(k)} \frac{w_{kj}}{2}(\mathbf{R}_k + \mathbf{R}_j)(\boldsymbol{t}_k - \boldsymbol{t}_j). \tag{4}$$

We also need to consider $\frac{\partial G(T')}{\partial \lambda_i} = 0$, which gives us the constraints in Eq. 1. Then the whole system of equations can be compactly written as

$$\boldsymbol{M}\boldsymbol{x} = \boldsymbol{b}, \tag{5}$$

where $\boldsymbol{M}$ is composed of the discrete Laplace-Beltrami operator and the barycentric coordinates, $\boldsymbol{x}$ is composed of the deformed positions $\boldsymbol{t}'$ and the newly-introduced variables $\boldsymbol{\lambda}$, and $\boldsymbol{b}$ is a vector composed of the right-hand side expression from Eq. 5 and the triangular mesh vertex positions $\boldsymbol{v}'$. The linear system can be solved by some common methods, such as the Newton's method.

2.2. Calculation of Barycentric Coordinates

In this section, we discuss how to quickly calculate the barycentric coordinates of sampled points in a tetrahedron. Consider a single tetrahedron with vertices $V_i(x_i, y_i, z_i)$ $(i = 1, 2, 3, 4)$, and the query sampled point $q(x, y, z)$, the barycentric coordinates of q can be calculated as:

$$bc_i = \text{Det}_i / \text{Det}_0, \tag{6}$$

where

$$\text{Det}_0 = \begin{vmatrix} x_1 & y_1 & z_1 & 1 \\ x_2 & y_2 & z_2 & 1 \\ x_3 & y_3 & z_3 & 1 \\ x_4 & y_4 & z_4 & 1 \end{vmatrix}, \tag{7}$$

and Det_i can be obtained by replacing the i-th row of Det_0 with coordinates of the query point, e.g.

$$\text{Det}_1 = \begin{vmatrix} x & y & z & 1 \\ x_2 & y_2 & z_2 & 1 \\ x_3 & y_3 & z_3 & 1 \\ x_4 & y_4 & z_4 & 1 \end{vmatrix} \tag{8}$$

The query point q lies in the tetrahedron if and only if $0 < bc_i < 1, i \in \{1, 2, 3, 4\}$.

There are about 32 million query points and thousands of tetrahedrons to render a 400×400 image, which is time-consuming. Therefore, we simplify the query process by only considering the tetrahedrons containing vertices near the query point. Note that different vertices are shared by different numbers of tetrahedrons. In order to calculate them efficiently in a consistent matrix formulation, we fill extra tetrahedrons as placeholders to make the dimension consistent between different query points, explained in Alg. 1.

Algorithm 1 Determine the tetrahedron for query point

Input

cage vertices V: [NV,3]	vertex position
cage tetrahedrons T: [NT,4]	vertex index
query point q: [1,3]	point position
(NV: #. vertices;	NT: #. tetrahedrons)

Preprocess:

1: Determine the max degree of tetrahedron vertices, denoted as D.
2: Construct a map $V2T$: [NV,D] from vertex index to a tetrahedron index list, which contains the vertex. The list is filled to length D with -1.

Calculation:

1: Find the K nearest vertices $N(q)$ for q
2: Extract D tetrahedrons with $V2T$ for each v $\in N(q)$
3: Iterate the the D*K tetrahedrons to determine the tetrahedral q belongs to.
4: Return the tetrahedron index and corresponding barycentric coordinates according to Eq. 6. If none of them contains q, returns index -1 and barycentric coordinates [0,0,0,0].

2.3. Self-Captured Datasets

In order to demonstrate the capability of our method, we captured several real-world scenes. For each scene, we took 2-3 circles around the center object and selected a certain number of images to train the NeRF network. The number of training images for each scene is around 150∼200. They are sampled uniformly, i.e. one out of every three due to high video fps. In addition, we use an extra image for validation in each scene.

2.4. Training & Inference Details

Our approach does not require additional manipulation during the training phase. We train the neural radiance field network following the NeRF [4] configuration, which costs 10∼12h on an Nvidia 2080Ti GPU. During the inference stage, this approach needs to determine the tetrahedron where the sampling point is located and its barycentric coordinates. The rendering time of a 400×400 image is in the interval of 40∼50s, up and down depending on the number of tetrahedrons in the tetrahedral cage. We adopt the training strategy of NeRF++ [9], which divides the space into the object centric near view and the far view. The reconstructed mesh will only contain the near view, which is then simply scissored to obtain the stand-alone target object. During the editing, the user only needs to specify some control points and then drag the control points or specify new coordinates for the control points to deform the triangular mesh. The efforts the user should take to edit the mesh is the same as ARAP [6].

3. Additional Editing Results

In this section, we show more editing results on both synthetic data and real captured scenes, as illustrated in Figs. 1-3 respectively. For synthetic data, we use "toad" from NSVF [3] and two characters from mixamo [2]. The "toad" can be edited from a static posture to a jumping posture. The two articulated characters illustrate that our method can also be applied to the NeRF of articulated objects, especially human bodies. For real captured scenes, except for the horse statue from FVS dataset [5] and the teddy bear from BlendedMVS [8], the rest are captured by ourselves. It can be seen that we can change the posture of the object. For example, two snakes turn offensive, the horse statue can change into a leap posture, and the bear raises its hand. We can also edit NeRF of other man-made objects. For example, we can bend a Roman stone column into an arch, enlarge the wooden cabinet, and distort the sports equipment.

4. Intermediate Results

To illustrate that our NeRF editing approach is faithful to the user's editing, we visualize the intermediate results, including the triangular mesh before editing and after editing and the corresponding tetrahedral meshes. The visualization results are shown in Fig. 4.

We further show the edited triangular mesh and tetrahedral mesh in the ablation study of "Necessity of edit on triangular mesh" in Fig. 6.

5. Application: Deformation transfer results

As mentioned in the main paper, we can also adopt deformation transfer methods in our approach in addition to interactive user editing. Here we show an example where we can transfer the movements of a human face from a video clip to a head sculpture, as shown in Fig. 5. Specifically, we first use DECA [1] to reconstruct the mesh sequence of the face from the input video, then use the deformation transfer method [7] to transfer the movements of the mesh sequence to the extracted mesh from the sculpture NeRF network, and further use our method to generate image synthesis results. It can be seen that the sculpture in the results can well reproduce the movements of the human face.

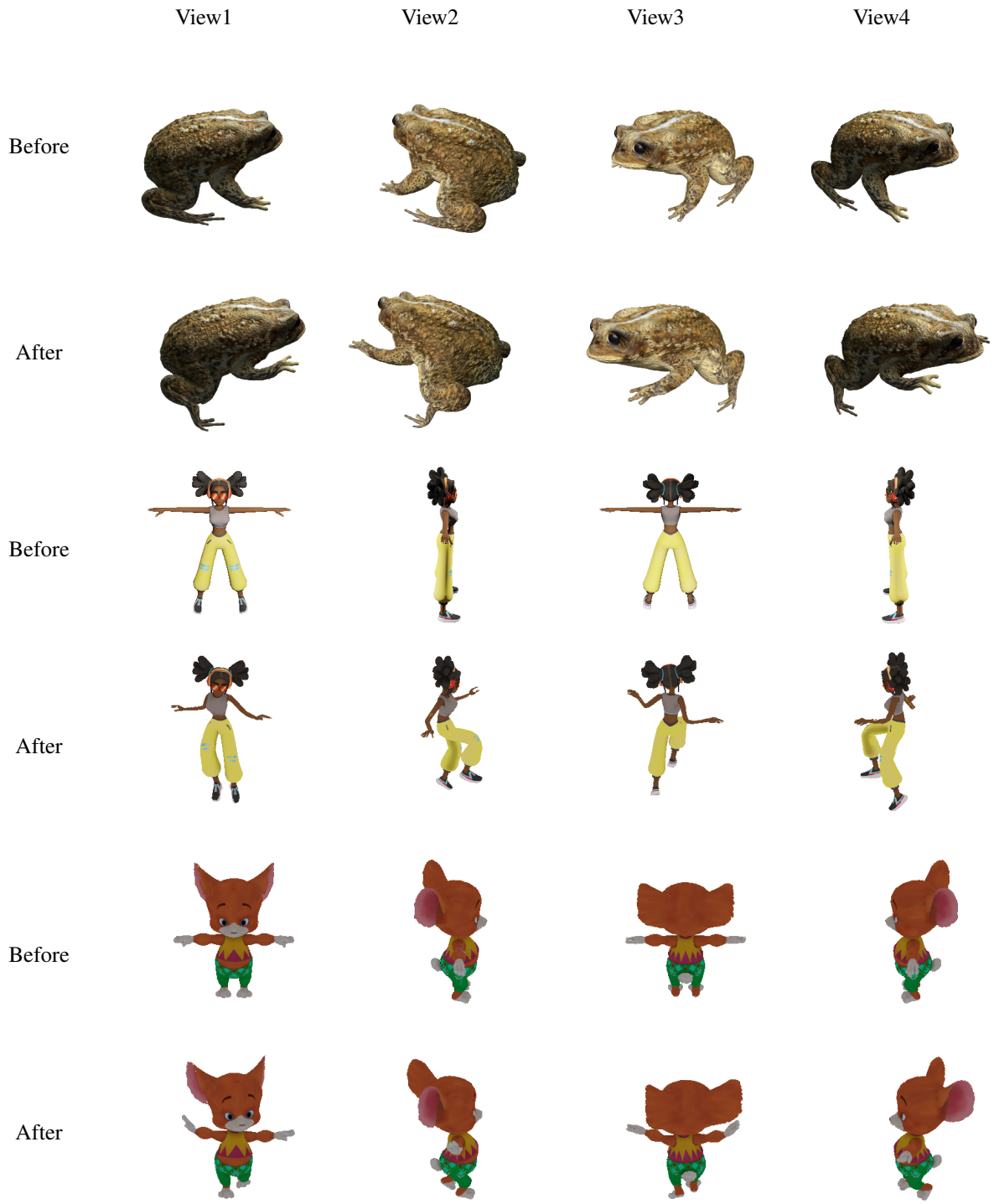

Figure 1. We show the editing results (row “After”) compared with NeRF rendering results (row “Before”) on synthetic data under different views. Different columns show different views.

References

[1] Yao Feng, Haiwen Feng, Michael J Black, and Timo Bolkart. Learning an animatable detailed 3D face model from in-the-wild images. *ACM Transactions on Graphics (TOG)*, 40(4):1–13, 2021. 3

[2] Adobe Inc. Mixamo. https://www.mixamo.com. 3

[3] Lingjie Liu, Jiatao Gu, Kyaw Zaw Lin, Tat-Seng Chua, and Christian Theobalt. Neural sparse voxel fields. *Advances in Neural Information Processing Systems*, 33, 2020. 3

[4] Ben Mildenhall, Pratul P Srinivasan, Matthew Tancik, Jonathan T Barron, Ravi Ramamoorthi, and Ren Ng. NeRF: Representing scenes as neural radiance fields for view synthesis. In *European Conference on Computer Vision*, pages 405–421. Springer, 2020. 3

[5] Gernot Riegler and Vladlen Koltun. Free view synthesis. In *European Conference on Computer Vision*, pages 623–640. Springer, 2020. 3

[6] Olga Sorkine-Hornung and Marc Alexa. As-rigid-as-possible surface modeling. In *Symposium on Geometry Processing*, 2007. 1, 3

[7] Robert W Sumner and Jovan Popović. Deformation transfer for triangle meshes. *ACM Transactions on Graphics (TOG)*, 23(3):399–405, 2004. 3

[8] Yao Yao, Zixin Luo, Shiwei Li, Jingyang Zhang, Yufan Ren, Lei Zhou, Tian Fang, and Long Quan. BlendedMVS: A large-scale dataset for generalized multi-view stereo networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020. 3

[9] K. Zhang, G. Riegler, Noah Snavely, and V. Koltun. NeRF++: Analyzing and improving neural radiance fields. *ArXiv*, abs/2010.07492, 2020. 3

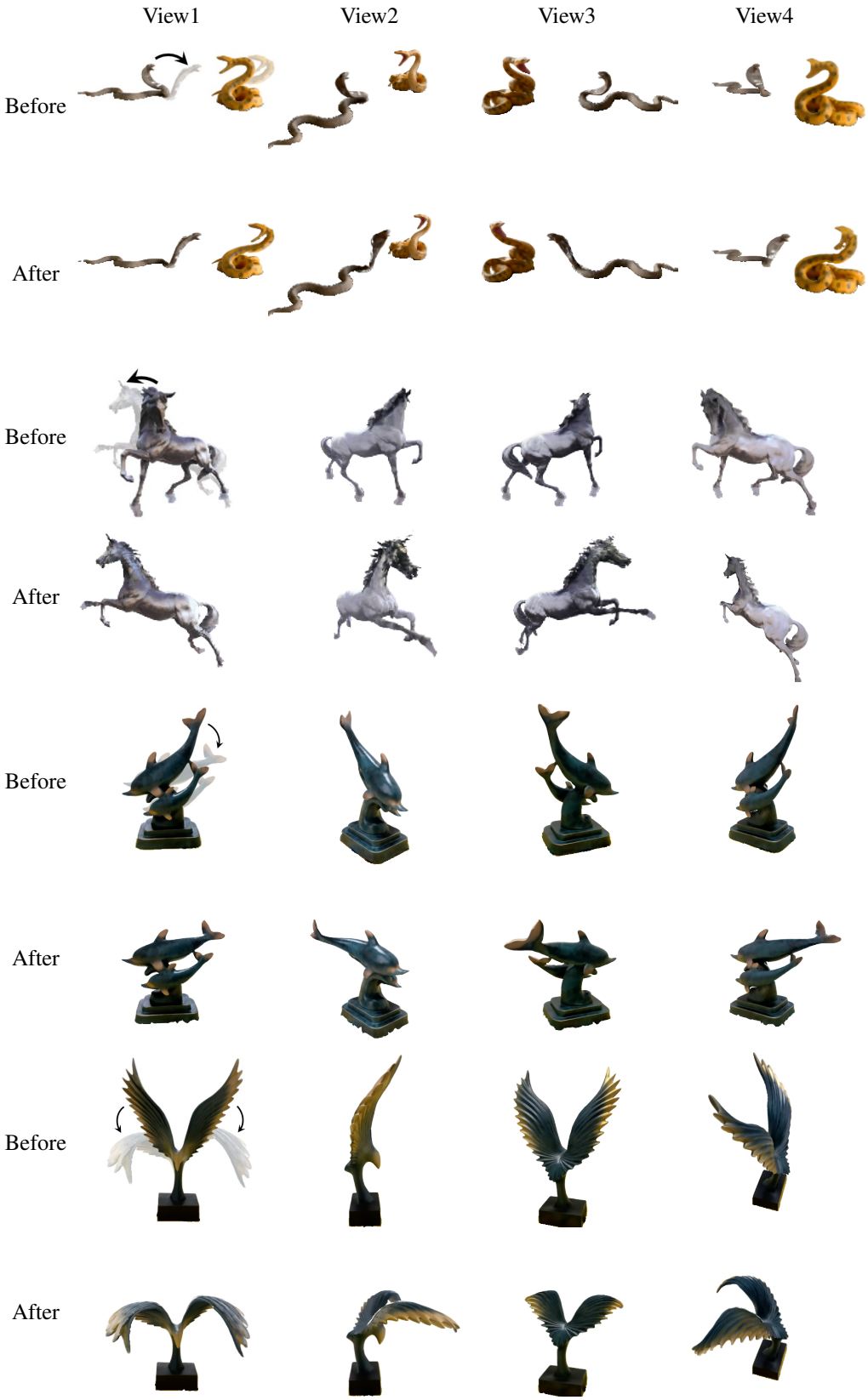

Figure 2. Results of our NeRF editing (row "After") compared with original NeRF results (row "Before") on the captured data. Different columns show different views. We edit the static neural radiance fields and exhibit the deformed results under different views.

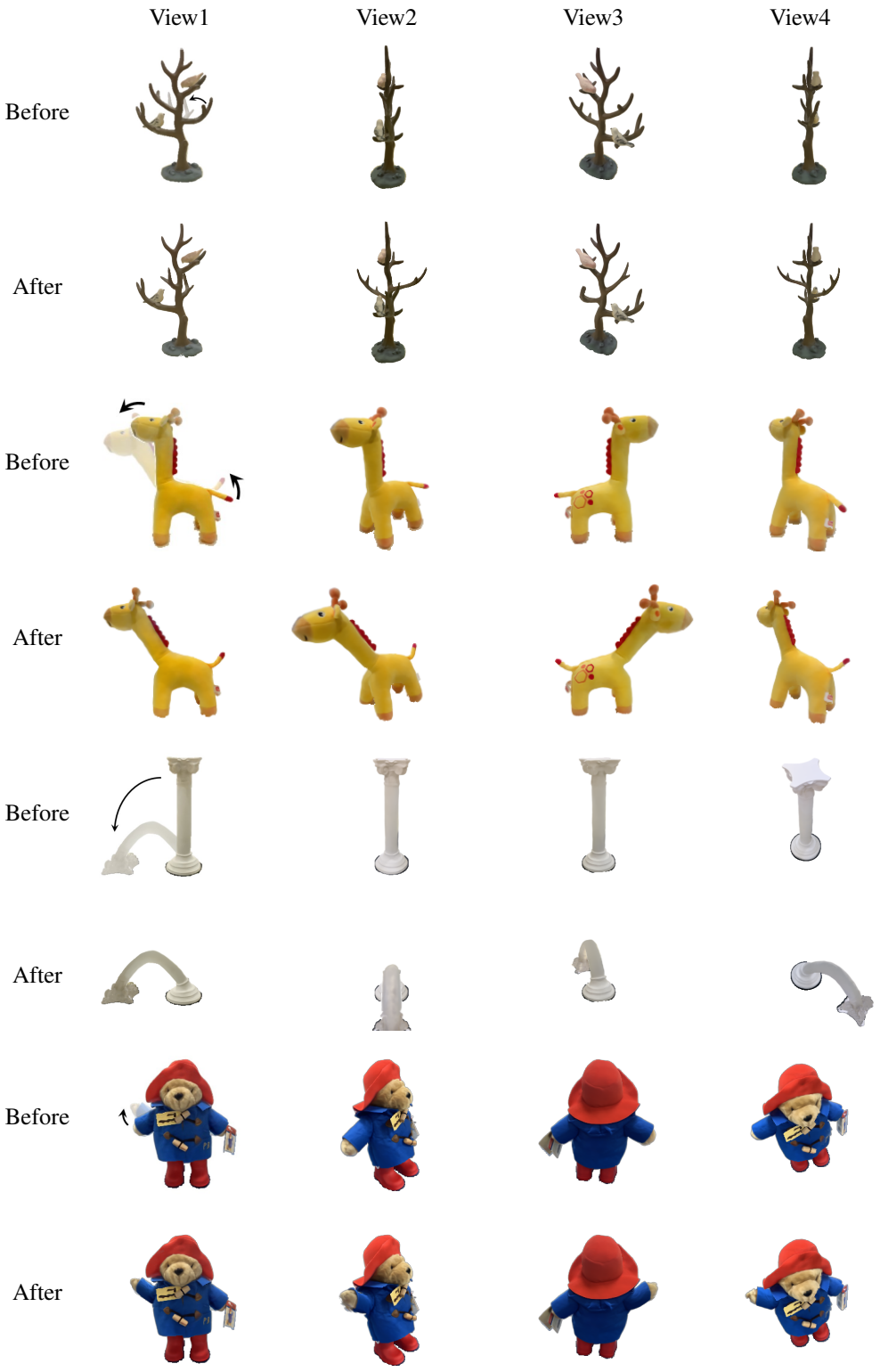

Figure 3. Results of our NeRF editing (row "After") compared with original NeRF results (row "Before") on the captured data. Different columns show different views. We edit the static neural radiance fields and exhibit the deformed results under different views.

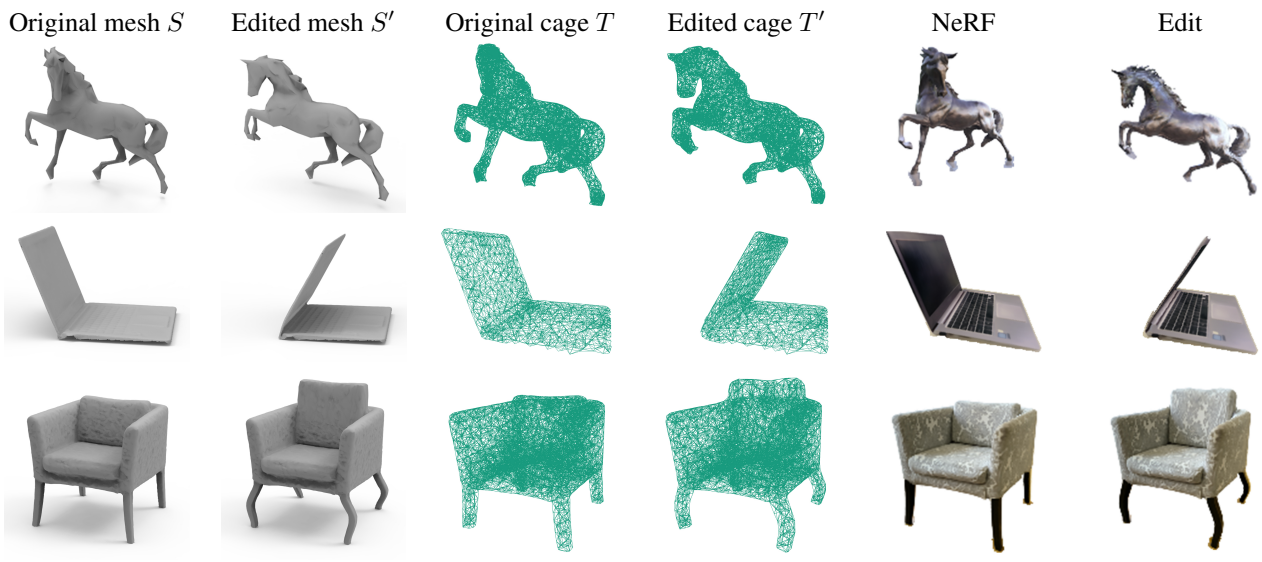

Figure 4. Visualizations of the intermediate results which illustrate that our NeRF editing approach is faithful to the user's editing.

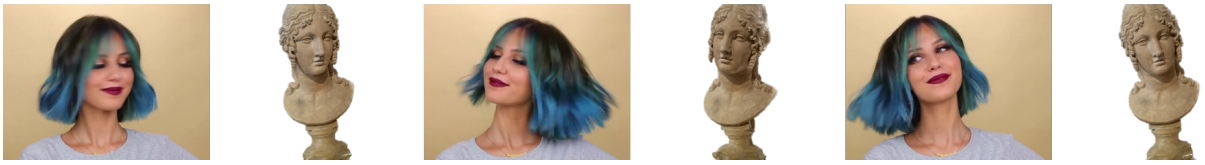

Figure 5. We show an application of our method. Given a video clip, we can adjust the sculpture's head to align with the given video frames with deformation transfer.

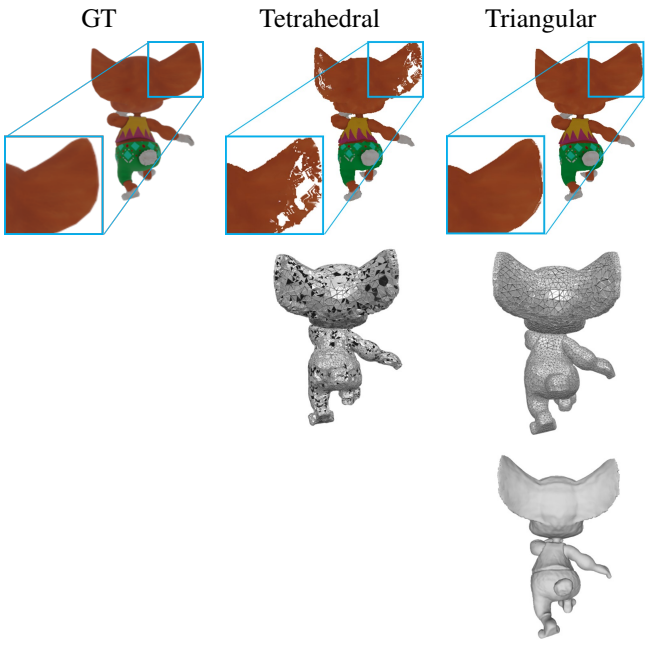

Figure 6. Ablation study of editing on the tetrahedral mesh or triangular mesh. It can be seen that editing on the tetrahedral mesh will bring in artifacts in rendered results. We also show the intermediate results, including the edited triangular mesh and tetrahedral mesh.