EditableNeRF: Editing Topologically Varying Neural Radiance Fields by Key Points

Chengwei Zheng Wenbin Lin Feng Xu
School of Software and BNRist, Tsinghua University

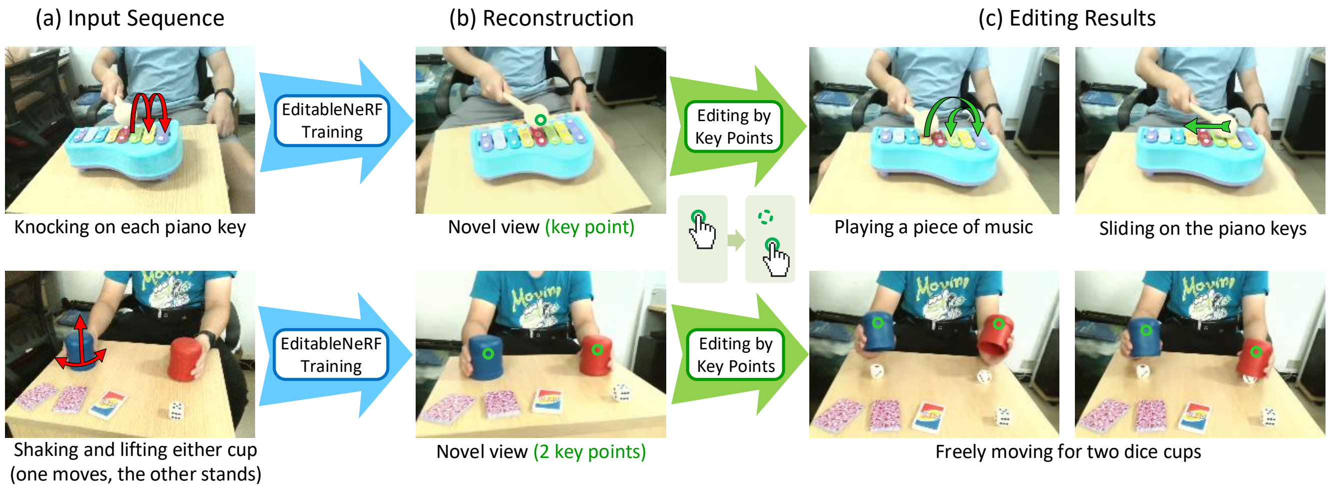
(a) Input Sequence
(b) Reconstruction
(c) Editing Results
EditableNeRF Training
Editing by Key Points
Knocking on each piano key
Novel view (key point)
Playing a piece of music
Sliding on the piano keys
Shaking and lifting either cup (one moves, the other stands)
Novel view (2 key points)
Freely moving for two dice cups

Figure 1. Taking an image sequence (a) as input, EditableNeRF is trained fully automatically to reconstruct the captured scene (b) and can handle topological changes. After training, end-users are able to edit the scene (c) by controlling the automatically picked-out key points (circled in green in (b)). Our method enables multi-dimensional editing and can generate novel scenes that are unseen during training.

Abstract

Neural radiance fields (NeRF) achieve highly photo-realistic novel-view synthesis, but it's a challenging problem to edit the scenes modeled by NeRF-based methods, especially for dynamic scenes. We propose editable neural radiance fields that enable end-users to easily edit dynamic scenes and even support topological changes. Input with an image sequence from a single camera, our network is trained fully automatically and models topologically varying dynamics using our picked-out surface key points. Then end-users can edit the scene by easily dragging the key points to desired new positions. To achieve this, we propose a scene analysis method to detect and initialize key points by considering the dynamics in the scene, and a weighted key points strategy to model topologically varying dynamics by joint key points and weights optimization. Our method supports intuitive multi-dimensional (up to 3D) editing and can generate novel scenes that are unseen in the input sequence. Experiments demonstrate that our method achieves high-quality editing on various dynamic scenes and outperforms the state-of-the-art. We will release our code and captured data.

1. Introduction

Neural radiance fields (NeRF) [23] have shown great power in novel-view synthesis and enable many applications as this method achieves photo-realistic rendering [9]. Recent techniques have further improved NeRF by extending it to handle dynamic scenes [27, 30, 40] and even topologically varying scenes [28]. However, these works mainly focus on reconstruction itself but do not consider scene editing. Thus, for rendering, only the camera views can be changed, while the modeled scenes cannot be edited.

Recently, some frameworks have been proposed to make neural radiance fields editable in different aspects. Some of them aim to edit the reconstructed appearance and enable relighting [2, 35, 54]; some allow controlling the shapes and colors of objects from a specific category [15, 20, 44, 47]; and some divide the scene into different parts and the location of each part can be modified [48, 49, 52]. However, the dynamics of moving objects cannot be edited by the previous methods. And this task becomes much more challenging when the dynamics contain topological changes. Topological changes can lead to motion discontinuities (e.g., between the hammer and the piano keys, between the cups and the table in Fig. 1) in 3D space and further cause noticeable artifacts if they are not modeled well. A state-of-the-art framework CoNeRF [16] tries to resolve this problem by using manual supervision. However, it only supports limited and one-dimensional editing for each scene part, requiring user annotations as supervision.

We propose EditableNeRF, editable topologically varying neural radiance fields that are trained without manual supervision and support intuitive multi-dimensional (up to three-dimensional) editing. The key of our method is to represent motions and topological changes by the movements of some sparse surface key points. Each key point is able to control the topologically varying dynamics of a moving part, as well as other effects like shadow and reflection changes through the neural radiance fields. This key-point-based method enables end-users to edit the scene by easily dragging the key points to their desired new positions.

To achieve this, we first apply a scene analysis method to detect key points in the canonical space and track them in the full sequence for key point initialization. We introduce a network to estimate spatially-varying weights for all scene points and use the weighted key points to model the dynamics in the scene, including topological changes. In the training stage, our network is trained to reconstruct the scene using the supervision from the input image sequence, and the key point positions are also optimized by taking motion (optical flow) and geometry (depth maps) constraints as additional supervision. After training, the scene can be edited by controlling the key points' positions, and novel scenes that are unseen during training can also be generated.

The contribution of this paper lies in the following aspects:

- Key-point-driven neural radiance fields achieving intuitive multi-dimensional editing even with topological changes, without requiring annotated training data.
- A weighted key points strategy modeling topologically varying dynamics by joint key points and weights optimization.
- A scene analysis method to detect and initialize key points by considering the dynamics in the scene.

2. Related Work

2.1. Novel-View Synthesis

Many methods achieve rendering novel-view images by reconstructing scenes and objects into meshes [5, 6, 12, 39, 55], neural voxels [21, 33], and multi-plane images [7, 24, 58]. Besides these methods based on discrete representations, some methods also achieve novel-view synthesis by using continuous representations [26, 34] and have shown great potential in this task.

Neural radiance fields (NeRF) [23] achieve photo-realistic rendering in novel-view synthesis by leveraging continuous implicit functions of density and view-dependent color to represent static scenes. To handle dynamic scenes, time-variant latent codes could be used to encode time-variant components based on NeRF, but requiring multi-view video inputs [17]. To further enable dynamic reconstructions from a single-view sequence, deformation fields implemented by MLPs are applied to warp objects in each frame into a canonical space [27, 30, 40]. Some methods also utilize estimated depth maps [45], ToF depth images [1], or optical scene flow [18] to improve the performance of dynamic neural radiance fields. HyperNeRF [28] further extends dynamic NeRF to reconstruct topologically varying scenes by modeling canonical spaces with different topology states into a unified continuous hyperspace, and the discontinuous deformations in 3D space caused by topological changes can be modeled by continuous functions in hyperspace. However, unlike traditional explicit representations such as triangular meshes, NeRF-based methods represent the scenes by implicit functions, making the modeled scenes difficult to be edited.

2.2. Editing on Neural Radiance Fields

As our method focuses on NeRF editing, we mainly discuss NeRF-based methods that support user editing in this section. For editing on explicit representations or other implicit representations, please refer to [4, 32, 50, 57].

One approach to edit neural radiance fields is to segment the scene into different components and build MLP for each component [48, 49, 52]. Assuming that different components are individual, this representation allows control of the placements and the relative positions of these components, as well as deleting or reduplicating a component. But these methods do not support editing the dynamics inside a component and only result in limited applications.

In addition, some methods achieve relighting and material editing on neural fields [2, 35, 54] by decomposing the scene into surface normals, lights, albedo, and material. Texture editing can also be accomplished by a 3D-to-2D texture mapping [46].

Besides, there are also some methods that focus on modeling a specific category of objects [15, 20, 44, 47] instead of general objects. A common solution for this problem is to model the category of objects with conditional NeRF and use latent codes as conditions to encode the variations of different objects in this category. Then the shape and appearance can be edited by changing the latent codes or by fine-tuning the network [20], and even controlled by

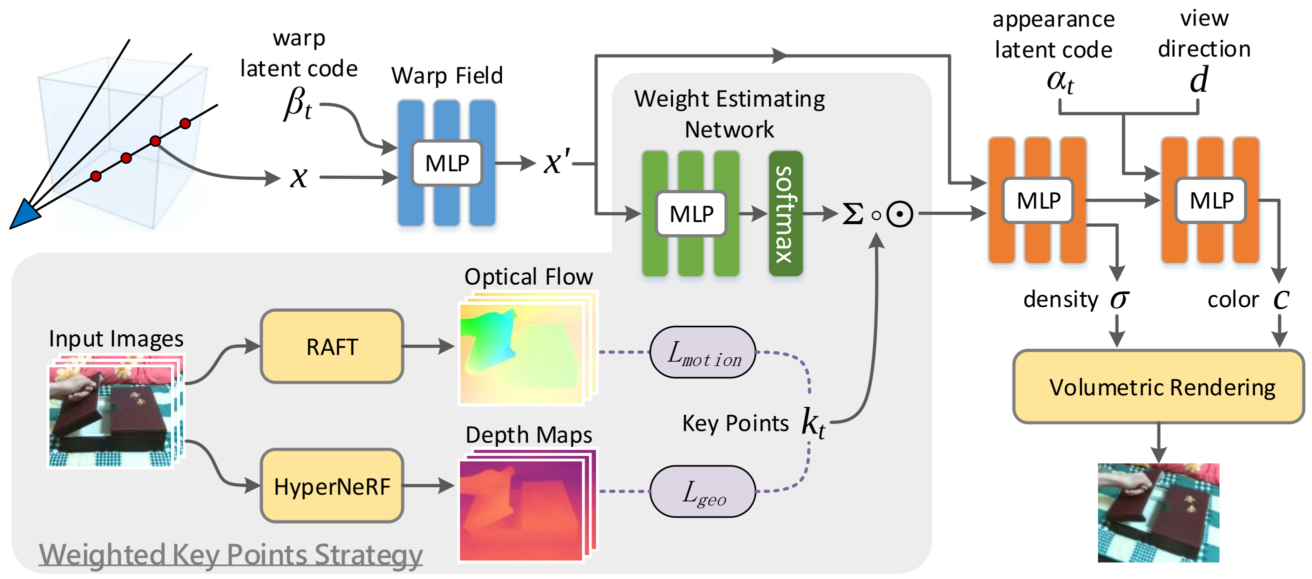

Figure 2. EditableNeRF pipeline. The query point x is first warped into the canonical space by a warp field and a latent code β_t in frame t. Next, we compute the key point weights of this canonical point x' and use it to calculate a linear combination of all key point positions k_t, called *weighted key points*. After that, we feed the following NeRF MLP with the weighted key points and x', then the output density and color are used for volumetric rendering. In the training stage, optical flow and depth maps are used to supervise key point positions.

text [41] with the help of a multi-modal model. And many methods also focus on modeling editable human bodies or human faces based on NeRF representation. By using human body parametric models and skinning techniques such as SMPL [22], neural radiance fields have been extended to model the human body and can be animated by controlling skeleton poses [3, 19, 25, 29, 36]. Human face parametric models also contribute to extending NeRF for human face modeling and controlling [8, 10, 14, 37, 42, 56], and even driven by audio [13]. However, general objects cannot be handled by these methods.

Recently, NeRF-editing [51] proposes to deform NeRF on static objects by extracting explicit meshes, deforming the meshes, and transferring the deformations back into the implicit representations. However, this method cannot handle dynamic scenes. CoNeRF [16] proposes an attribute re-rendering method based on dynamic NeRF. This method requires users to provide annotations in several frames, including masks for every dynamic part and their corresponding attribute values, for network training. Then these parts could be edited by controlling one-dimensional attribute values. We will show our advantages against CoNeRF in Sec. 4.2.

3. EditableNeRF

Input with color image sequence, our method can reconstruct the captured scene fully automatically based on neural radiance fields, and the topologically varying dynamics are modeled using surface key points. After reconstruction, end-users can edit the scene by controlling the key points.

Our pipeline is shown in Fig. 2. First, We apply two methods, HyperNeRF [28] and RAFT [38], to derive the depth maps of input frames and optical flow between adjacent input images, respectively. Then we apply a scene analysis method to detect and initialize key points for each frame (Sec. 3.3). After that, our NeRF-based network (Sec. 3.1) can be trained fully automatically (Sec. 3.2) to model the captured scene based on our weighted key points strategy. When the reconstruction is finished, the reconstructed scene can be edited by dragging the key points to desired positions (Sec. 3.4).

3.1. Network

We first introduce our network architecture, which is shown in Fig. 2. Our network represents the scene as a field of density and radiance [23]. Given a query point, similarly to other dynamic NeRF methods [27,28], we first use a warp field to model slight topology-preserving motions.

$$x' = T(x, \beta_t). \tag{1}$$

Here the warp filed T maps the query 3D point x to its canonical location x', and β_t is the warp latent code in frame t. Note that as discussed in [28], it's hard for this continuous warp field to model discontinuous movements caused by topological changes.

Then we need to model different topology and motion states in the canonical space. We find that motions and topological changes are always related to some movements

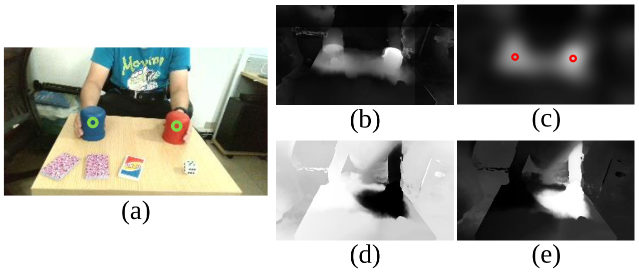

Figure 3. Some examples related to key points. (a) shows an input frame and its corresponding key points (circled in green). (b) is a 2D version of the variances of ambient coordinates, and (c) shows selected key points based on (b) after a 2D Gaussian filter. (d) and (e) demonstrate the weights of two key points, respectively. These weights are obtained by using the surface points corresponding to the pixels as query points. (a)-(e) are in the same viewpoint.

of surface points, so we achieve this modeling by making use of sparse surface key points. These 3D key points are attached to the objects' surfaces and also move with the objects. Each key point is able to control the topologically varying dynamics of a moving part and also some effects like shadow and reflection changes. An example of key points is shown in (a) of Fig. 3. For each moving part in the scene, we automatically select one corresponding key point, which will be detailed in Sec. 3.3, and the number of key points is denoted as N. The key points' positions in each input frame will be optimized automatically in our training stage to achieve this modeling.

We assume that different locations in the canonical space are affected by different key points. So for a query point x', an MLP followed by softmax is used to decide which key point should control its dynamics. We call this network *weight estimating network*, which takes canonical coordinate x' as input and outputs a weight vector $w \in \mathbb{R}^N$, indicating how each key point affects the query point x'.

$$w = W(x'). \tag{2}$$

An example of these spatially-varying key point weights is shown in (d) and (e) of Fig. 3.

We then construct a weighted key points vector p by taking a linear combination of all key point positions k to model the topologically varying dynamics of x'.

$$p_t(x') = \sum_{0 \le i < N} w^i(x') \cdot k_t^i. \tag{3}$$

The superscript i is the index of key points, and the subscript t is the frame index. If there is only one object that moves and causes topological changes, our method will model this scene with only one key point ($N = 1$), and (3) becomes $p_t(x') = k_t$ because the softmax always outputs a scalar 1. So we directly set $p_t(x')$ to be k_t in this situation.

Next, the 3D canonical coordinate x' and the weighted key points p are concatenated to construct a coordinate in hyperspace for topologically varying scene modeling. This hyperspace is proposed in HyperNeRF [28]. In addition to 3D space, HyperNeRF makes use of ambient dimensions to model objects in hyperspace, and different topology states are encoded with different ambient coordinates. Discontinuous deformations caused by topological changes in 3D space can be modeled by continuous functions (such as MLP) in hyperspace, and more details can be found in [28]. Here we use the weighted key points p as ambient coordinates, modeling topologically varying dynamics.

Finally, the following NeRF MLP is fed with this 6D coordinate in hyperspace:

$$(c, \sigma) = H(x' \oplus p_t(x'), d, \alpha_t). \tag{4}$$

Here d is the view direction, and α_t is the appearance latent code as in [28]. This NeRF MLP outputs the color c and the density σ that can be used in volumetric rendering. To render an image, we should trace the camera rays of all pixels, sample points along these rays, obtain their colors and densities, and run the volumetric rendering, which are the same as in the original NeRF [23].

3.2. Loss Functions and Training

All the latent codes and MLP parameters are optimized in the training stage to model the scene. As we use key points to encode topologically varying dynamics, we need to additionally optimize key point positions in each input frame. To keep our key points on the object surfaces and to be time-consistent, novel losses are added in our training stage.

First, we propose a motion loss, which constrains that the key points in two adjacent frames should be consistent with the optical flow from pre-trained RAFT [38].

$$L_{motion}(t,i) = \left\| \Pi_{t+1}(k_{t+1}^i) - \Pi_t(k_t^i) - F_t^{t+1}(\Pi_t(k_t^i)) \right\|^2, \tag{5}$$

where Π_t is the projection function using the input camera pose of frame t, and F_t^{t+1} is the optical flow from frame t to frame $t+1$. This loss ensures that the projected key points in different frames correspond to the same surface point.

The motion loss provides good supervision in 2D image space, while the key points are in the 3D space. Thus, a geometry loss can help to keep the key points on the object surfaces.

$$L_{geo}(t,i) = \left\| \Phi_t(k_t^i) - D_t(\Pi_t(k_t^i)) \right\|^2. \tag{6}$$

Here the function $\Phi_t(k_t^i)$ calculates the distance from the key point k_t^i to the camera position in frame t, and D_t denotes the depth map rendered from HyperNeRF [28] in the original camera view. This HyperNeRF is pre-trained before our training stage and takes the same input as ours.

Besides, we also apply the reconstruction loss between the rendered RGB images C and the input images $\widetilde{C}$, as well as the warp regularization loss to make sure that the warp field only models slight movements, avoiding the ambiguity between the warp field and the weighted key points model.

$$L_{rec}(t) = \left\| C_t(k_t, \alpha_t, \beta_t) - \widetilde{C}_t \right\|^2, \tag{7}$$

$$L_{reg}(t) = \frac{1}{\|S_t\|} \sum_{x \in S_t} \|x - T(x, \beta_t)\|^2, \tag{8}$$

where S_t is the set of surface points in frame t.

3.3. Key Point Detection and Initialization

To initialize the network training, we need to determine the key point number N and obtain the initial 3D locations of key points. To achieve this, we apply a scene analysis method, which first finds reference key points in canonical space and reference frames where these reference key points are on the object surfaces, then initializes key points' positions in each frame.

As our key points are used to model topologically varying dynamics, we find the 3D points in the canonical space with dramatic dynamics including topological changes as our reference key points. Recall that in HyperNeRF [28], the ambient dimensions are used to model motion and topology states. And we have already trained a HyperNeRF for depth maps in (6). So we can detect key points by making use of the ambient dimensions of this pre-trained HyperNeRF. We find the positions with the locally greatest variations of ambient coordinates, and use them as our reference key points.

To be specific, for each input frame, we trace the original camera rays of all pixels and find the corresponding surface points. A 3D voxel volume in the canonical space is then built to record the ambient coordinates of these surface points. After traversing the whole input sequence, we compute the variance of ambient coordinates for every voxel, followed by a 3D Gaussian filter. And the center points of the voxels with local maximum variances after Gaussian blur will be selected as the reference key points k_{ref}. As our 3D version is difficult to visualize, we show a 2D version of this process in (b) and (c) of Fig. 3 by rendering all frames in a fixed camera view and computing the variance of each pixel, followed with a 2D Gaussian filter.

Then for each reference key point, we need to select a reference frame in which the reference key point is on the object surface. We use a similar formulation with the geometry loss in (6) to decide this. (Here we omit the index of key points as they are handled individually.)

$$\|\Phi_t(k_{ref}) - D_t(\Pi_t(k_{ref}))\|^2 < \delta, \tag{9}$$

where δ is a pre-defined threshold. The first frame t that satisfies (9) will be selected as the reference frame t_{ref}.

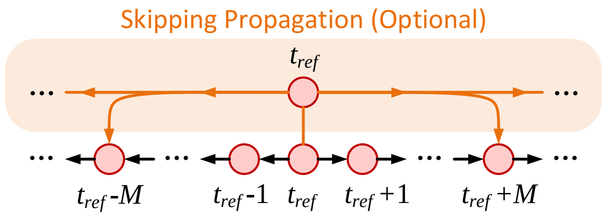

Figure 4. Propagating the reference key point in the reference frame to other frames for initialization. Skipping propagation is only used for some long input sequences.

Now for each key point, we have a reference key point position and a corresponding reference frame. To initialize key point positions in the whole sequence, we propagate this reference key point to other frames by optical flow from pre-trained RAFT [38]. The reference key point is first projected into the input image of the reference frame to get its 2D position, and the 2D position is propagated frame by frame using optical flow as shown in Fig. 4. Then these 2D positions are projected back into 3D space by depth maps from HyperNeRF. Note that there are accumulative errors in this initialization due to frame-by-frame propagation, while these errors will be eliminated in our training stage.

For some very long input sequences, we found that the initialization method above may not perform well. This is because the accumulative errors may become too large, and the key points in the image space may be propagated into other objects (e.g., background). So we propose a skipping propagation method as shown in Fig. 4. For each key point, we additionally propagate the reference key point in the reference frame every M frames (i.e., t_{ref} to $t_{ref} + M$, t_{ref} to $t_{ref} + 2M$, and so on), and replace the frame-by-frame positions if their confidences are greater than a threshold. This confidence is calculated by the consistency between the forward optical flow and the backward optical flow:

$$Conf(t) = \left\| F_t^{t_{ref}}(F_{t_{ref}}^t(\widehat{k}_{ref})) - \widehat{k}_{ref} \right\|^{-1}, \tag{10}$$

where $t = t_{ref} + iM$, $i \in \mathbb{Z}$, and the hat of $\widehat{k}_{ref}$ indicates that it is a 2D position in the reference frame.

3.4. Editing by Key Points

After training, users can easily edit the modeled scenes by feeding the network with desired key point positions. As the key points are in the 3D space, our method supports up to three-dimensional editing for each part. We also provide a graphical user interface (GUI) in Sec. 4.4.

4. Experiments

We show some results after editing in Fig. 5. And please refer to our accompanying video and supplementary materials for more results and details.

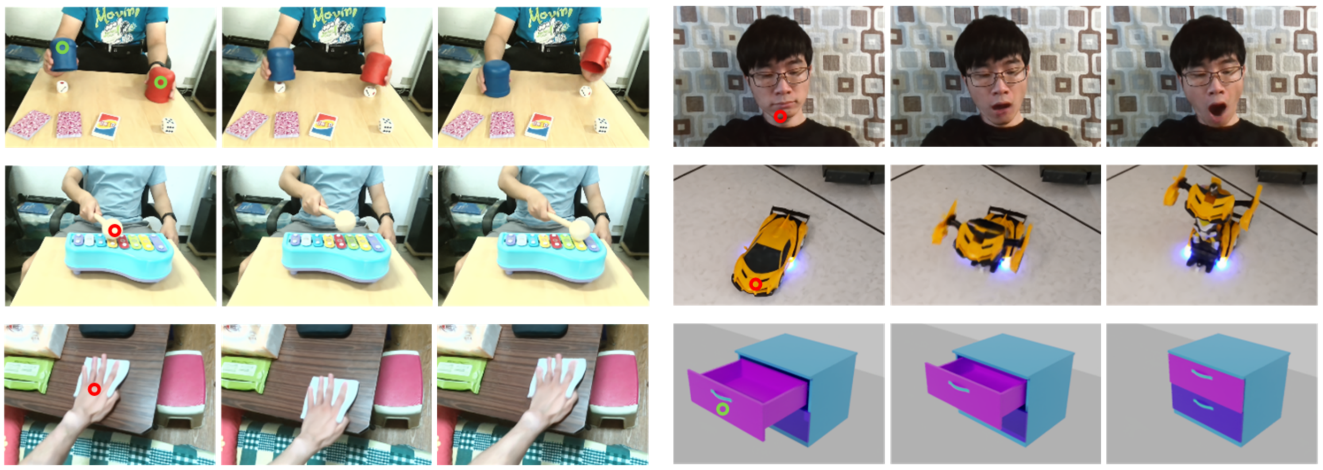

Figure 5. Our editing results on various scenes. The first image of each scene also shows the key points (circled in red or green). Results on the left side contain multi-dimensional editing. The input data of the 2nd row on the right side is provided by CoNeRF [16]. The last row on the right side is from a synthetic sequence. Note that some effects, like shadow and reflection changes, are also edited correctly.

4.1. Implementation Details

We set the weight of motion loss to 10^{-4}, the weight of geometry loss to 0.5, and the weight of warp regularization loss to 0.1. The real data is captured in a resolution of 1280×720 and down-sampled to 320×180 for network training. Our network is trained on 4 NVIDIA GeForce RTX 3090 graphics cards, and takes around 5 hours for training with 250k iterations. Our code is based on HyperNeRF [28]. And the camera poses of input frames are solved by COLMAP [31].

Local coordinate systems for key points. The key point positions are transferred into local coordinate systems for normalization. For each key point, this local coordinate system takes the average key point position as the origin and scales the coordinates to ensure that the largest range of the three dimensions is 1, both based on the initialized positions. Without this normalization, the positional encoding functions [23] for key points will almost become linear functions when the positions vary in a small range. Besides, for some objects with complex geometries, our method may select duplicate key points on the same object. We can remove duplicate key points if the initialized local coordinates of two key points are always very similar to each other in the full sequence.

4.2. Comparisons

Here we compare our method with state-of-the-art methods HyperNeRF [28] and CoNeRF [16]. HyperNeRF is capable of topologically varying scene reconstruction but does not enable scene editing. CoNeRF allows topologically varying editing but only supports one-dimensional editing for each dynamic part, and user annotations, including masks for every part and their attribute values, are necessary for its pipeline. For example, to train a CoNeRF network on an opening mouth sequence, users have to select some input frames, mask the mouth regions in these frames, and set the corresponding attribute values to 1 when the mouth is open and -1 when the mouth is closed.

Qualitative results. As our method focuses on editing, we first evaluate the editing ability of our method. We compare our method with CoNeRF, while HyperNeRF does not support editing.

We show some editing results of ours and CoNeRF in Fig. 6. Firstly, as shown in the (a) results, our method based on 3D key points enables up to three-dimensional editing, while CoNeRF fails to encode multi-dimensional dynamics by using one-dimensional attribute values. Secondly, our training stage is fully automatic without user annotations. Especially when different parts are close to each other, it becomes quite difficult for end-users to provide very accurate masks at the boundaries, which further leads to artifacts in CoNeRF as shown in (c) of Fig. 6. In contrast, ours can distinguish different parts automatically. Besides, our editing method allows users to drag the key points to their desired positions, which is more intuitive than inputting attribute values as in CoNeRF.

Quantitative results on synthetic data. Here we compare our method with HyperNeRF and CoNeRF on synthetic data. The data sequence is synthesized by Kubric [11] and contains 400 frames for training. We also show some results on this sequence in Fig. 5. For CoNeRF training, we annotate 5% frames in the training set using ground truth masks and ground truth attribute values, which are the same as the experiment settings in CoNeRF paper [16]. While for our method, we still use the optical flow from RAFT and depth maps from HyperNeRF. We *do not* use ground

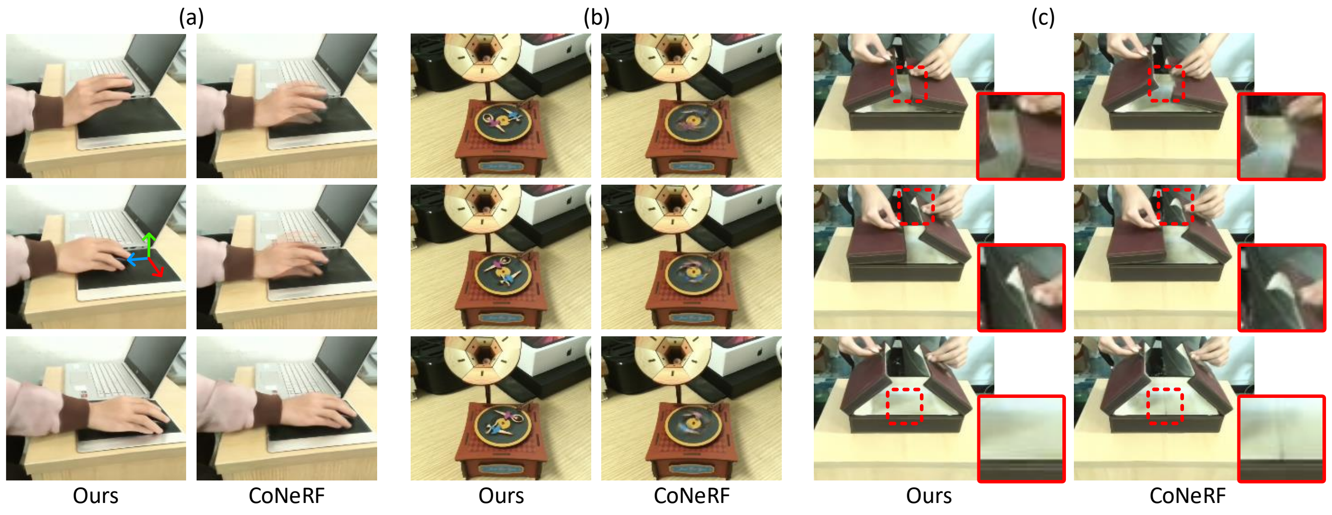

Figure 6. Qualitative comparisons with CoNeRF [16]. Our method does not require user annotations for training and supports multi-dimensional editing. Note that the rotations in (b) also cannot be represented by the one-dimensional attribute values in CoNeRF.

Method	Reconstruction PSNR ↑	Reconstruction MS-SSIM ↑	Reconstruction LPIPS ↓	Editing PSNR ↑	Editing MS-SSIM ↑	Editing LPIPS ↓
HyperNeRF [28]	43.03	0.9971	0.0826	-	-	-
CoNeRF [16]	40.82	0.9908	0.0932	39.80	0.9887	0.0958
Ours	**44.74**	**0.9978**	**0.0815**	**40.05**	**0.9962**	**0.0831**

Table 1. Quantitative comparisons of reconstruction and editing qualities on synthetic data. The reconstruction qualities are measured by the errors in novel-view synthesis. We report PSNR, MS-SSIM [43], and LPIPS [53]. Our method performs the best.

truth optical flow or ground truth depth maps, keeping these settings the same as for real data. Both compared methods are implemented by the original authors and are trained with the same batch size and iteration step as ours.

First, we compare the reconstruction qualities of these methods. We render the same synthetic scene in novel viewpoints as ground truth and evaluate the novel-view synthesis abilities. As shown in Table 1, our method reaches the best performance. Note that our method even slightly outperforms HyperNeRF on this task. This is because our method models the scene using key points, which makes it easier for our network to integrate the information from the frames with similar key point positions but in different viewpoints.

Next, we compare the editing qualities of our method and CoNeRF, while HyperNeRF cannot be used for editing. We derive ground truth key point positions and attribute values from ground truth motions and input them for our method and CoNeRF, respectively. Then compute the errors between the ground truth images and the rendered images after editing. Our method also outperforms CoNeRF as shown in Table 1.

Quantitative results on real data. As it is difficult to get novel-view ground truth for real data sequences, we turn to compare the interpolation qualities on real data. For a real data sequence with $2N$ frames, we pick out N frames with even indices as the training set, and the other N frames with odd indices are used as ground truth for testing. Table 2 demonstrates that all the compared methods get similar quantitative results on this task. Key point positions for our method, attribute values for CoNeRF, and all the latent codes are interpolated in this task. In CoNeRF training, we select 1% frames with extreme attribute values for annotations, as recommended by CoNeRF [16].

Method	PSNR ↑	MS-SSIM ↑	LPIPS ↓
HyperNeRF [28]	30.56	0.9864	0.1281
CoNeRF [16]	30.65	0.9869	0.1307
Ours	30.67	0.9869	0.1314

Table 2. Quantitative comparisons of interpolation qualities on real data.

Besides, when rendering the same real scene with two dynamic parts in the same resolution, CoNeRF takes 1.77s, HyperNeRF takes 0.95s, while ours takes 0.90s. And CoNeRF needs to add a new MLP for each dynamic part, which makes its network not as compact as the other methods.

Method	PSNR ↑	MS-SSIM ↑	LPIPS ↓
Base (w/o supervision)	24.45	0.8529	0.1702
+ L_{motion}	29.98	0.9485	0.1138
+ L_{motion} + L_{geo}	32.58	0.9674	0.1043
+ L_{motion} + L_{geo} + init	**40.05**	**0.9962**	**0.0831**

Table 3. Ablation Studies. We evaluate the motion loss, the geometry loss, and the initialization stage. Our final method in the last row performs the best.

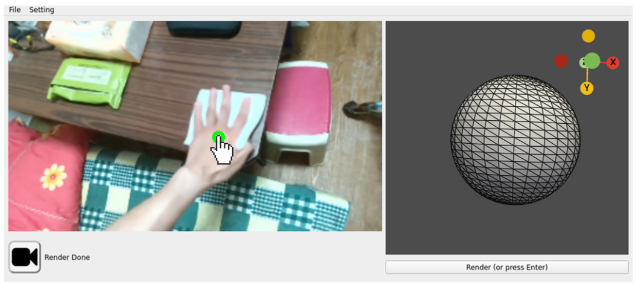

Figure 7. Graphical user interface. The left widget shows the rendered image and the corresponding draggable key points. The right widget allows the user to change the viewpoint. Our GUI also supports recording an editing sequence then rendering a video.

4.3. Ablation Studies

Our method makes use of 3D key points with the help of 2D optical flow and 1D depth maps. They are first utilized to initialize key point positions, then to formulate the motion loss and the geometry loss. We evaluate the two losses and the initialization stage in Table 3 by measuring the editing qualities on synthetic data. The base method does not use any information from optical flow or depth maps, and its modeled scene changes randomly according to key point movements, while our final method in the last row of Table 3 reaches the best performance.

4.4. Applications

Graphical user interface. We implement a graphical user interface (GUI) for editing and novel-view synthesis, which is shown in Fig. 7 and in our accompanying video. Note that end-users actually drag the key points in the 2D interface, so we provide 1D default depth values for key points to form 3D positions, and we also allow end-users to further edit these depth values. The default depth value for a pixel position is obtained by finding K closest key point positions in the input sequence after projecting into the current view and computing their average depth.

Novel scenes generation. Novel scenes that are unseen in the training sequence can also be generated by our method. For example, in the piano toy sequence of Fig. 1, the input data only contains knocking on each piano key, while our method can generate sliding on the piano keys by interpolation. And our method can also combine various dynamics of different parts to create novel scenes, such as the dice cups sequence results shown in Fig. 1.

Motion transfer. Once reconstructed, our modeled scenes can be driven by motions from other sequences. We show a phonograph toy driven by a disk in our supplementary video.

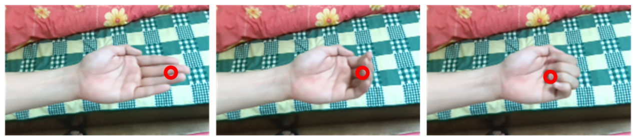

Figure 8. Editing results on a challenging scene where the selected key point is not always visible in the full sequence.

4.5. Discussions

We build our framework based on surface key points. While in some challenging sequences, there may not exists a proper surface point that is visible in all frames to become a key point. Our method can still get plausible results on these sequences, but the consistency of key points is not as good as in other scenes, as shown in Fig. 8.

Limitations. We assume that the dynamics of a canonical location mainly depend on one key point. If the scene becomes very complex that does not satisfy this assumption (e.g., a dancing human), our method may fail. Also, it's hard for our method to pick out surface key points for semi-transparent objects like smoke. Extrapolation cannot be performed well for our method when the key points are dragged too far away from their positions in the training sequence. Our method supports multi-dimensional editing, but if the captured objects only have one-dimensional dynamics (e.g., drawer only moves in 1D), our method can only generate one-dimensional dynamics. Besides, our method cannot work well when RAFT or HyperNeRF fails. We leave these problems as future works.

5. Conclusions

We propose EditableNeRF, editable topologically varying neural radiance fields that enable end-users to easily edit dynamic scenes. The key to achieving this is to build our framework by leveraging weighted key points to model topologically varying dynamics, which further achieves intuitive multi-dimensional editing. And a scene analysis method that can measure the dynamics in the scene is also proposed to detect and further initialize these key points. Our method is trained fully automatically using a single-view input sequence and can be easily used by end-users, bringing new applications for editable photo-realistic novel-view synthesis.

References

[1] Benjamin Attal, Eliot Laidlaw, Aaron Gokaslan, Changil Kim, Christian Richardt, James Tompkin, and Matthew O'Toole. Törf: Time-of-flight radiance fields for dynamic scene view synthesis. *Advances in neural information processing systems*, 34:26289–26301, 2021. 2

[2] Mark Boss, Raphael Braun, Varun Jampani, Jonathan T Barron, Ce Liu, and Hendrik Lensch. Nerd: Neural reflectance decomposition from image collections. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 12684–12694, 2021. 1, 2

[3] Jianchuan Chen, Ying Zhang, Di Kang, Xuefei Zhe, Linchao Bao, Xu Jia, and Huchuan Lu. Animatable neural radiance fields from monocular rgb videos. *arXiv preprint arXiv:2106.13629*, 2021. 3

[4] Yu Deng, Jiaolong Yang, and Xin Tong. Deformed implicit field: Modeling 3d shapes with learned dense correspondence. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10286–10296, 2021. 2

[5] Mingsong Dou, Philip Davidson, Sean Ryan Fanello, Sameh Khamis, Adarsh Kowdle, Christoph Rhemann, Vladimir Tankovich, and Shahram Izadi. Motion2fusion: Real-time volumetric performance capture. *ACM Transactions on Graphics (TOG)*, 36(6):1–16, 2017. 2

[6] Ruofei Du, Ming Chuang, Wayne Chang, Hugues Hoppe, and Amitabh Varshney. Montage4d: interactive seamless fusion of multiview video textures. In *Proceedings of the ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games*, pages 1–11, 2018. 2

[7] John Flynn, Michael Broxton, Paul Debevec, Matthew DuVall, Graham Fyffe, Ryan Overbeck, Noah Snavely, and Richard Tucker. Deepview: View synthesis with learned gradient descent. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2367–2376, 2019. 2

[8] Guy Gafni, Justus Thies, Michael Zollhofer, and Matthias Nießner. Dynamic neural radiance fields for monocular 4d facial avatar reconstruction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8649–8658, 2021. 3

[9] Kyle Gao, Yina Gao, Hongjie He, Denning Lu, Linlin Xu, and Jonathan Li. Nerf: Neural radiance field in 3d vision, a comprehensive review. *arXiv preprint arXiv:2210.00379*, 2022. 1

[10] Philip-William Grassal, Malte Prinzler, Titus Leistner, Carsten Rother, Matthias Nießner, and Justus Thies. Neural head avatars from monocular rgb videos. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 18653–18664, 2022. 3

[11] Klaus Greff, Francois Belletti, Lucas Beyer, Carl Doersch, Yilun Du, Daniel Duckworth, David J Fleet, Dan Gnanapragasam, Florian Golemo, Charles Herrmann, et al. Kubric: A scalable dataset generator. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3749–3761, 2022. 6

[12] Kaiwen Guo, Feng Xu, Tao Yu, Xiaoyang Liu, Qionghai Dai, and Yebin Liu. Real-time geometry, albedo, and motion reconstruction using a single rgb-d camera. *ACM Transactions on Graphics (ToG)*, 36(4):1, 2017. 2

[13] Yudong Guo, Keyu Chen, Sen Liang, Yong-Jin Liu, Hujun Bao, and Juyong Zhang. Ad-nerf: Audio driven neural radiance fields for talking head synthesis. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 5784–5794, 2021. 3

[14] Yang Hong, Bo Peng, Haiyao Xiao, Ligang Liu, and Juyong Zhang. Headnerf: A real-time nerf-based parametric head model. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 20374–20384, 2022. 3

[15] Wonbong Jang and Lourdes Agapito. Codenerf: Disentangled neural radiance fields for object categories. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 12949–12958, 2021. 1, 2

[16] Kacper Kania, Kwang Moo Yi, Marek Kowalski, Tomasz Trzciński, and Andrea Tagliasacchi. Conerf: Controllable neural radiance fields. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 18623–18632, 2022. 2, 3, 6, 7

[17] Tianye Li, Mira Slavcheva, Michael Zollhoefer, Simon Green, Christoph Lassner, Changil Kim, Tanner Schmidt, Steven Lovegrove, Michael Goesele, Richard Newcombe, et al. Neural 3d video synthesis from multi-view video. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5521–5531, 2022. 2

[18] Zhengqi Li, Simon Niklaus, Noah Snavely, and Oliver 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*, pages 6498–6508, 2021. 2

[19] Lingjie Liu, Marc Habermann, Viktor Rudnev, Kripasindhu Sarkar, Jiatao Gu, and Christian Theobalt. Neural actor: Neural free-view synthesis of human actors with pose control. *ACM Transactions on Graphics (TOG)*, 40(6):1–16, 2021. 3

[20] Steven Liu, Xiuming Zhang, Zhoutong Zhang, Richard Zhang, Jun-Yan Zhu, and Bryan Russell. Editing conditional radiance fields. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 5773–5783, 2021. 1, 2

[21] Stephen Lombardi, Tomas Simon, Jason Saragih, Gabriel Schwartz, Andreas Lehrmann, and Yaser Sheikh. Neural volumes: learning dynamic renderable volumes from images. *ACM Transactions on Graphics (TOG)*, 38(4):1–14, 2019. 2

[22] Matthew Loper, Naureen Mahmood, Javier Romero, Gerard Pons-Moll, and Michael J Black. Smpl: A skinned multi-person linear model. *ACM transactions on graphics (TOG)*, 34(6):1–16, 2015. 3

[23] B Mildenhall. Nerf: Representing scenes as neural radiance fields for view synthesis. In *European conference on computer vision*, 2020. 1, 2, 3, 4, 6

[24] Ben Mildenhall, Pratul P Srinivasan, Rodrigo Ortiz-Cayon, Nima Khademi Kalantari, Ravi Ramamoorthi, Ren Ng, and

Abhishek Kar. Local light field fusion: Practical view synthesis with prescriptive sampling guidelines. *ACM Transactions on Graphics (TOG)*, 38(4):1–14, 2019. 2

[25] Atsuhiro Noguchi, Xiao Sun, Stephen Lin, and Tatsuya Harada. Neural articulated radiance field. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 5762–5772, 2021. 3

[26] Jeong Joon Park, Peter Florence, Julian Straub, Richard Newcombe, and Steven Lovegrove. Deepsdf: Learning continuous signed distance functions for shape representation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 165–174, 2019. 2

[27] Keunhong Park, Utkarsh Sinha, Jonathan T Barron, Sofien Bouaziz, Dan B Goldman, Steven M Seitz, and Ricardo Martin-Brualla. Nerfies: Deformable neural radiance fields. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 5865–5874, 2021. 1, 2, 3

[28] Keunhong Park, Utkarsh Sinha, Peter Hedman, Jonathan T Barron, Sofien Bouaziz, Dan B Goldman, Ricardo Martin-Brualla, and Steven M Seitz. Hypernerf: a higher-dimensional representation for topologically varying neural radiance fields. *ACM Transactions on Graphics (TOG)*, 40(6):1–12, 2021. 1, 2, 3, 4, 5, 6, 7

[29] Sida Peng, Junting Dong, Qianqian Wang, Shangzhan Zhang, Qing Shuai, Xiaowei Zhou, and Hujun Bao. Animatable neural radiance fields for modeling dynamic human bodies. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 14314–14323, 2021. 3

[30] Albert Pumarola, Enric Corona, Gerard Pons-Moll, and Francesc Moreno-Noguer. D-nerf: Neural radiance fields for dynamic scenes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10318–10327, 2021. 1, 2

[31] Johannes L Schonberger and Jan-Michael Frahm. Structure-from-motion revisited. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4104–4113, 2016. 6

[32] Vincent Sitzmann, Julien Martel, Alexander Bergman, David Lindell, and Gordon Wetzstein. Implicit neural representations with periodic activation functions. *Advances in Neural Information Processing Systems*, 33:7462–7473, 2020. 2

[33] Vincent Sitzmann, Justus Thies, Felix Heide, Matthias Nießner, Gordon Wetzstein, and Michael Zollhofer. Deepvoxels: Learning persistent 3d feature embeddings. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2437–2446, 2019. 2

[34] Vincent Sitzmann, Michael Zollhöfer, and Gordon Wetzstein. Scene representation networks: Continuous 3d-structure-aware neural scene representations. *Advances in Neural Information Processing Systems*, 32, 2019. 2

[35] Pratul P Srinivasan, Boyang Deng, Xiuming Zhang, Matthew Tancik, Ben Mildenhall, and Jonathan 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*, pages 7495–7504, 2021. 1, 2

[36] Shih-Yang Su, Frank Yu, Michael Zollhöfer, and Helge Rhodin. A-nerf: Articulated neural radiance fields for learning human shape, appearance, and pose. *Advances in Neural Information Processing Systems*, 34:12278–12291, 2021. 3

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

[38] Zachary Teed and Jia Deng. Raft: Recurrent all-pairs field transforms for optical flow. In *European conference on computer vision*, pages 402–419. Springer, 2020. 3, 4, 5

[39] Justus Thies, Michael Zollhöfer, and Matthias Nießner. Deferred neural rendering: Image synthesis using neural textures. *ACM Transactions on Graphics (TOG)*, 38(4):1–12, 2019. 2

[40] Edgar Tretschk, Ayush Tewari, Vladislav Golyanik, Michael Zollhöfer, Christoph Lassner, and Christian 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*, pages 12959–12970, 2021. 1, 2

[41] Can Wang, Menglei Chai, Mingming He, Dongdong Chen, and Jing Liao. Clip-nerf: Text-and-image driven manipulation of neural radiance fields. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3835–3844, 2022. 3

[42] Ting-Chun Wang, Arun Mallya, and Ming-Yu Liu. One-shot free-view neural talking-head synthesis for video conferencing. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10039–10049, 2021. 3

[43] Zhou Wang, Eero P Simoncelli, and Alan C Bovik. Multiscale structural similarity for image quality assessment. In *The Thrity-Seventh Asilomar Conference on Signals, Systems & Computers, 2003*, volume 2, pages 1398–1402. Ieee, 2003. 7

[44] Fangyin Wei, Rohan Chabra, Lingni Ma, Christoph Lassner, Michael Zollhöfer, Szymon Rusinkiewicz, Chris Sweeney, Richard Newcombe, and Mira Slavcheva. Self-supervised neural articulated shape and appearance models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 15816–15826, 2022. 1, 2

[45] Wenqi Xian, Jia-Bin Huang, Johannes Kopf, and Changil Kim. Space-time neural irradiance fields for free-viewpoint video. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9421–9431, 2021. 2

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

[47] Christopher Xie, Keunhong Park, Ricardo Martin-Brualla, and Matthew Brown. Fig-nerf: Figure-ground neural radiance fields for 3d object category modelling. In *2021 International Conference on 3D Vision (3DV)*, pages 962–971. IEEE, 2021. 1, 2

[48] Bangbang Yang, Yinda Zhang, Yinghao Xu, Yijin Li, Han Zhou, Hujun Bao, Guofeng Zhang, and Zhaopeng Cui.

Learning object-compositional neural radiance field for editable scene rendering. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 13779–13788, 2021. 1, 2

[49] Hong-Xing Yu, Leonidas Guibas, and Jiajun Wu. Unsupervised discovery of object radiance fields. In *International Conference on Learning Representations*, 2021. 1, 2

[50] Yu-Jie Yuan, Yu-Kun Lai, Tong Wu, Lin Gao, and Ligang Liu. A revisit of shape editing techniques: From the geometric to the neural viewpoint. *Journal of Computer Science and Technology*, 36(3):520–554, 2021. 2

[51] Yu-Jie Yuan, Yang-Tian Sun, Yu-Kun Lai, Yuewen Ma, Rongfei Jia, and Lin Gao. Nerf-editing: geometry editing of neural radiance fields. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 18353–18364, 2022. 3

[52] Jiakai Zhang, Xinhang Liu, Xinyi Ye, Fuqiang Zhao, Yanshun Zhang, Minye Wu, Yingliang Zhang, Lan Xu, and Jingyi Yu. Editable free-viewpoint video using a layered neural representation. *ACM Transactions on Graphics (TOG)*, 40(4):1–18, 2021. 1, 2

[53] Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 586–595, 2018. 7

[54] Xiuming Zhang, Pratul P Srinivasan, Boyang Deng, Paul Debevec, William T Freeman, and Jonathan T Barron. Nerfactor: Neural factorization of shape and reflectance under an unknown illumination. *ACM Transactions on Graphics (TOG)*, 40(6):1–18, 2021. 1, 2

[55] Chengwei Zheng and Feng Xu. Dtexfusion: Dynamic texture fusion using a consumer rgbd sensor. *IEEE Transactions on Visualization and Computer Graphics*, 2021. 2

[56] Yufeng Zheng, Victoria Fernández Abrevaya, Marcel C Bühler, Xu Chen, Michael J Black, and Otmar Hilliges. Im avatar: Implicit morphable head avatars from videos. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13545–13555, 2022. 3

[57] Zerong Zheng, Tao Yu, Qionghai Dai, and Yebin Liu. Deep implicit templates for 3d shape representation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1429–1439, 2021. 2

[58] Tinghui Zhou, Richard Tucker, John Flynn, Graham Fyffe, and Noah Snavely. Stereo magnification: learning view synthesis using multiplane images. *ACM Transactions on Graphics (TOG)*, 37(4):1–12, 2018. 2