Neural Articulated Radiance Field

Atsuhiro Noguchi[1] Xiao Sun[2] Stephen Lin[2] Tatsuya Harada[1,3]

[1]The University of Tokyo [2]Microsoft Research Asia [3]RIKEN

Abstract

We present Neural Articulated Radiance Field (NARF), a novel deformable 3D representation for articulated objects learned from images. While recent advances in 3D implicit representation have made it possible to learn models of complex objects, learning pose-controllable representations of articulated objects remains a challenge, as current methods require 3D shape supervision and are unable to render appearance. In formulating an implicit representation of 3D articulated objects, our method considers only the rigid transformation of the most relevant object part in solving for the radiance field at each 3D location. In this way, the proposed method represents pose-dependent changes without significantly increasing the computational complexity. NARF is fully differentiable and can be trained from images with pose annotations. Moreover, through the use of an autoencoder, it can learn appearance variations over multiple instances of an object class. Experiments show that the proposed method is efficient and can generalize well to novel poses. We make the code, model and demo available for research purposes at https://github.com/nogu-atsu/NARF.

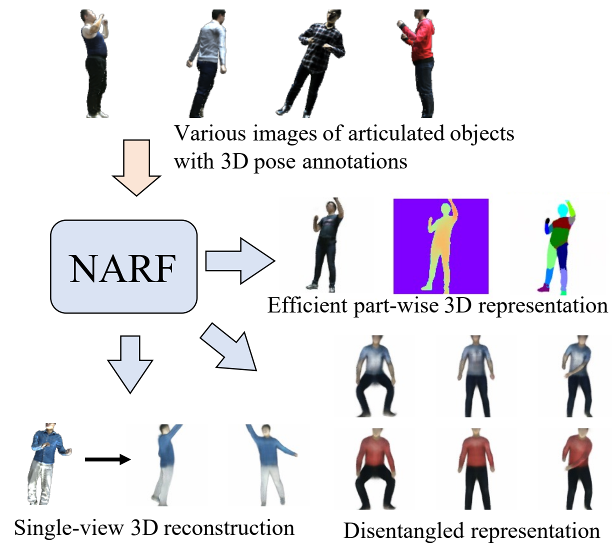

Figure 1. NARF is an efficient pose-aware 3D representation trained from only pose-annotated images. The learned representation is part-based and disentangles appearance properties.

1. Introduction

In this work, we aim to learn a representation for rendering novel views and poses of 3D articulated objects, such as human bodies, from images. Our approach follows the inverse graphics paradigm [28, 27, 38, 29] of analyzing an image by attempting to synthesize it with compact graphics codes. These codes are typically disentangled to allow for rendering of scenes/objects with fine-grained control over individual appearance properties such as object location, pose, lighting, texture, and shape. For the case of humans, synthesis of novel views and poses can be useful for applications such as movie making, photo editing, virtual clothing [65, 30] and motion transfer [34, 4].

Various inverse graphics based approaches have been specifically designed for static scenes [61, 20, 55], rigid objects [9, 69, 63, 57, 3], blend shapes for keypoints [62, 4] and dense meshes [65, 34]. However, efficient deformation modeling of articulated 3D objects using neural networks remains a challenging task due to the large variance of joint locations (especially for endpoints such as hands), severe self-occlusions, and high non-linearity in forward kinematic transformations [70]. Though work has been done to enable explicit control over the underlying human pose [34] and key point locations [4], their neural rendering methods are either limited to 2D [4], which prevents modeling of view-dependent appearance [7], or based on mesh representations [34], where rendering quality can be affected by the resolution of the discrete template mesh.

Recent progress on the implicit representation of 3D objects and scenes, such as signed distance functions [19, 43] and occupancy fields [12, 39], has greatly promoted the development of the inverse graphics paradigm. Such representations are lightweight in model size, continuous, and differentiable, making them highly practical in comparison with the previously-dominant volumetric representations [11, 16, 23, 40, 47, 56, 58, 68]. Particularly, Mildenhall et al. [41] propose the neural radiance field (NeRF) that takes a single continuous 5D coordinate (3D spatial location and 2D viewing direction) as input and outputs the volume density and view-dependent emitted radiance at each spatial location. Combined with a classical differentiable volume rendering technique [22], it is able to synthesize novel views by learning from a sparse set of input views of static scenes. NeRF completely discards the mesh-based representation and replaces it with the radiance-based model which can effectively and efficiently encode view-dependent appearance, enabling it to reproduce scenes of complex geometry with high fidelity.

In this paper, we extend NeRF to an Articulated-NeRF, called a Neural Articulated Radiance Field (NARF), to represent articulated 3D objects. Accounting for 3D articulation within the NeRF framework is a challenging problem because a complex, non-linear relationship exists between a kinematic representation of 3D articulations and the resulting radiance field, making it hard to model implicitly in a neural network [70]. In addition, the radiance field at a given 3D location is influenced by at most a single articulated part and its parents along the kinematic tree, while the full kinematic model is provided as input. As a result, dependencies of the output to irrelevant parts may inadvertently be learned, which is known to hurt model generalization to poses unseen in training [66].

To address these issues, we propose a method that predicts the radiance field at a 3D location based on only the most relevant articulated part. This part is identified using a set of sub-networks that output the probability for each part given the 3D location and the 3D geometric configuration of the parts. The spatial configurations of parts are computed *explicitly* with a kinematic model, rather than modeled implicitly in the network. A NARF then predicts the density and view-dependent radiance of the 3D location conditioned on the properties of only the selected part.

The presented NARF model has the following properties:

- It learns a disentangled representation of camera viewpoint, bone parameters, pose, and view-dependent color, allowing these appearance properties to be individually controlled in rendering.

- A dense 3D representation is learned from a sparse set of single-view 2D images with pose annotations of the articulated object, which could potentially be obtained through external pose estimation techniques on multi-view images with known camera parameters [21] or by incorporating pose estimation within the NARF framework.

- Part segmentation is learned from single-view 2D images with pose annotation. Additional supervision is not needed.

- NARF can be trained for articulated objects of various shape and appearance, through the use of an autoencoder that extracts a latent appearance vector.

With this approach, it becomes possible to render both novel views and poses of articulated 3D objects from pose annotated 2D images with little increase in computational complexity.

2. Related Work

Articulated 3D Shape Representations The deformation of articulated objects is traditionally modeled by skinning techniques [17, 18, 31, 32] in which the location of surface mesh vertices is determined from bone transformations controlled by the kinematics [2]. Effective skinning models with subtle pose-dependent and identity-dependent deformation modeling [35] have been developed for the human body. However, the representation capacity of skinning based models is limited to the resolution of the discrete template mesh, and sophisticated shading techniques [49] are usually required for high quality image rendering. In addition, a large amount of 3D scan data and expert supervision are required to prepare a template mesh.

Recently, Deng et al. [8] proposed a neural network based articulated shape representation (NASA). NASA learns the neural indicator/occupancy function [6, 39, 44] of every point in space, conditioned on a latent pose vector that encodes a piece-wise decomposition. NASA provides a continuous and differentiable representation for 3D articulated shapes. However, ground truth occupancy is required to train the network, and NASA does not learn appearance, a critical element for rendering.

Articulated Pose Conditioned Image Generation The recent advance of image generation models such as variational autoencoders (VAE) [26] and generative adversarial networks (GANs) [15] provides powerful tools for generating realistic-looking images. Image generation for articulated objects (typically persons) conditioned on target poses is an important direction with various applications like movie making, photo editing, virtual clothing [65, 30] and motion transfer [34, 4]. A majority of these works [4, 36, 37, 53, 59, 10, 1] generate the image of a person in a target pose by learning a GAN model from 2D keypoint maps of the target pose. The appearance information of this person is provided by explicit concatenation with an image of this person in the target pose [36, 53], automatically encoded for a single person [4] or using autoencoders [37]. These works are limited to 2D, which prevents modeling of view-dependent appearance [7]. Some works [65] leverage the underlying 3D mesh representation and transfer appearance from one mesh to another using aligned mesh triangles. The quality of a mesh based representation is bounded by the resolution of its discrete template mesh and a 3D template mesh is required.

Implicit 3D representation Our work builds on the recent success of the implicit 3D representation. This representation is memory efficient, continuous, and topology-free, and has been used for learning 3d shape [39, 44, 54], 3d texture [42], static scenes [41, 55], parts decomposition [14, 13], articulated objects [8], deformation [48, 33, 64, 45], 3d reconstruction from sparse images [50, 51], and image synthesis [52, 5].

Early methods required ground truth 3D geometry [39, 44], but in combination with differentiable rendering, they evolved to learn from 2D images. In particular, neural radiance fields (NeRF) [41] is capable of learning a 3D representation of complex scenes using only multi-view posed images. However, NeRF addresses static scenes and cannot handle deformable objects. Very recently, methods have been proposed to extend NeRF to learn deformations and dynamics [48, 33, 64, 45]. These models have been successful in learning deformable implicit representations using posed video frames [48, 33, 64, 45]. However, these models do not take into account the structure of the object, so they cannot generate images with explicit pose control.

3. Method

In this section, we present neural articulated radiance fields, a novel implicit representation for articulated 3D objects based on Neural Radiance Fields (NeRF) [41]. We start by briefly reviewing the basic NeRF formulation for static scenes in Sec. 3.1. In Sec. 3.2, NeRF is extended to be conditioned on pose via a kinematic model, and a straightforward baseline is derived from this. We reformulate the pose-conditioned NeRF to allow for rigid object transformations as well as global shape variations in Sec. 3.3. In Sec. 3.4, we represent articulated 3D objects as a composition of movable rigid object parts controlled by forward kinematic rules. To achieve constant model complexity with respect to the number of object parts, we propose an efficient *Disentangled NARF* architecture. The training strategy is then presented in Sec. 3.5.

3.1. Neural Radiance Field Revisited

A neural network is used to represent a Radiance Field such that 3D location $\mathbf{x} = (x, y, z)$ and 2D viewing direction $\mathbf{d}$ is converted to density σ and RGB color value c.

$$F_\Theta : (\mathbf{x}, \mathbf{d}) \to (c, \sigma) \tag{1}$$

F_Θ consists of positional encoding (PE) layers γ and two ReLU MLP networks. PE maps an input scalar into a higher dimensional space to represent high-frequency detail of the scene. Specifically, the volume density σ is a function of the location $\mathbf{x}$ only, while the RGB color c is a function of both location $\mathbf{x}$ and viewing direction $\mathbf{d}$.

$$F_{\Theta_\sigma} : (\gamma(\mathbf{x})) \to (\sigma, \mathbf{h}),\ F_{\Theta_c} : (\mathbf{h}, \gamma(\mathbf{d})) \to (c), \tag{2}$$

where $\mathbf{h}$ is a hidden feature vector.

Classical volume rendering [22] is used to render the color $\mathbf{C}(\mathbf{r})$ of a camera ray $\mathbf{r}(t) = \mathbf{o} + t\mathbf{d}$ with near and far bounds t_n and t_f, and where $\mathbf{o}$ denotes the camera position.

$$T(t) = \exp(-\int_{t_n}^{t} \sigma(\mathbf{r}(s))ds), \tag{3}$$

$$\mathbf{C}(\mathbf{r}) = \int_{t_n}^{t_f} T(t)\sigma(\mathbf{r}(t))c(\mathbf{r}(t), \mathbf{d})dt \tag{4}$$

$T(t)$ denotes the accumulated transmittance along the ray. The integrals are computed by a discrete approximation over sampled points along the ray $\mathbf{r}$.

$$T_j = \exp\left(-\sum_{k=1}^{j-1} \sigma_k \delta_k\right), \mathbf{C}(\mathbf{r}) = \sum_{j=1}^{N} T_j (1 - \exp(-\sigma_j \delta_j)) c_j \tag{5}$$

Here, σ_j and c_j are the density and color at the j^{th} point on the ray $\mathbf{r}$, and δ_j is the distance between the j^{th} and $(j+1)^{th}$ sample points.

NeRF is trained on images of a single static scene taken from multiple views with known camera parameters. The density and color of each location are trained so that the rendered image for each of the views becomes close to its ground truth. After training, high resolution images can be synthesized from any viewpoint.

3.2. Pose-Conditioned NeRF: A Baseline

Our goal is to extend the representation capacity of NeRF from static scenes to deformable articulated objects whose configurations can be described by a kinematic model [2]. The radiance field of a 3D location is thus conditioned on the pose configuration. Once this "pose-conditioned NeRF" is learned, novel poses can be rendered in addition to novel views, by changing the input pose configurations.

Kinematic Model Formally, the kinematic model [2] represents an articulated object of $P + 1$ joints, including endpoints, and P bones in a tree structure where one of the joints is selected as the root joint and each remaining joint is linked to its single parent joint by a bone of fixed length.

Specifically, the root joint J_0 is defined by a global position $\mathbf{p}$ and orientation $\mathbf{o}$. Let $\boldsymbol{\zeta}_i$ be the bone length from the i^{th} joint J_i to its parent, $i \in \{1, ...P\}$, and $\boldsymbol{\theta}_i$ denotes the rotation angles of the joint with respect to its parent joint.

A bone, considered as a rigid object, defines a local rigid transformation between a joint and its parent. The transformation matrix $\mathbf{T}^i_{local}$ is computed as

$$\mathbf{T}^i_{local} = \text{Rot}(\boldsymbol{\theta}_i)\text{Trans}(\boldsymbol{\zeta}_i), \tag{6}$$

where Rot and Trans are the rotation and translation matrix, respectively. The global transformation from the root joint to joint J_i can thus be obtained by multiplying the transformation matrices along the bones from the root joint to the i^{th} joint:

$$\mathbf{T}^i = \Pi_{k \in \text{Pa}(i)} \mathbf{T}^k_{local}, \mathbf{T}^0 = \text{Rot}(\mathbf{o})\text{Trans}(\mathbf{p}) \tag{7}$$

where $\text{Pa}(i)$ includes the i^{th} joint and all of its parent joints along the kinematic tree. The corresponding global rigid transformation $l^i = \{R^i, \mathbf{t}^i\}$ for the i^{th} joint can then be obtained from the transformation matrix $\mathbf{T}^i$.

Baseline The most straightforward way to condition the radiance field at a 3D location $\mathbf{x}$ on a kinematic pose configuration $\mathcal{P} = \{\mathbf{p}, \mathbf{o}, \boldsymbol{\zeta}, \boldsymbol{\theta}\}$ is to directly concatenate a vector representing $\mathcal{P}$ as the model input. Since the forward kinematic computation is a complex non-linear function [70] as is hard to simulate in the neural networks, we use the transformations $l^i = \{R^i, \mathbf{t}^i\}$ obtained by the forward kinematics as network inputs.

$$F^{\mathcal{P}}_{\Theta} : (\gamma(\mathbf{x}), \gamma(\{l^i | i = 1, ..., P\}), \gamma(\mathbf{d})) \rightarrow (\sigma, c) \tag{8}$$

We refer to this naive approach as Pose-conditioned NeRF (P-NeRF). The implementation details can be found in the supplemental material. Though P-NeRF establishes dependency between the radiance field and pose, generalization with this model is difficult because of the following two reasons.

- ***Implicit Transformations.*** An articulated object consists of several rigid bodies, and the particles on the object should move with the rigid transformations of the parts when the pose changes. Therefore, the movement of points can be explicitly described using rigid body transformations of each part, but such transformations may be difficult for a neural network to learn implicitly.

- ***Part Dependency.*** The existence of a particle at a 3D location depends only on the parameters of the bone it lies on and its parents along the kinematic tree. However, all the parameters are used to estimate the radiance field of a single location in Eq. 8. As the training on such 3D locations is backpropagated to all the parameters, the network may learn erroneous dependencies that do not physically exist. Correct pose predictions may still be obtained for test poses seen in the training data, but model generalization to novel poses may be degraded [66].

Towards addressing the above issues, we decompose the articulated object into P rigid object parts. Each part has its own local coordinate system defined by the rigid transformation $l^i = \{R^i, \mathbf{t}^i\}$, which is *explicitly* estimated using forward kinematics, rather than modeled implicitly by a neural network. Then, we show how a rigidly transformed object part can be effectively modeled in a rigidly transformed neural radiance field (RT-NeRF) in Section 3.3. Based on RT-NeRF, we describe in Section 3.4 how to train a single unified NeRF that encodes multiple parts in a manner that avoids the part dependency issue.

3.3. Rigidly Transformed Neural Radiance Field

Given a rigid transformation $l = \{R, \mathbf{t}\}$ of an object, we now estimate the radiance field in the object coordinate system where the density is constant with respect to a *local* 3D location. Formally,

$$F^l_{\Theta_\sigma} : (\gamma(\mathbf{x}^l)) \rightarrow (\sigma, \mathbf{h}) \tag{9}$$

where $\mathbf{x}^l = R^{-1}(\mathbf{x} - \mathbf{t})$ represents the 3D location in the local object coordinate system.

We expect the model to handle certain shape variations. For example, the limb length and thickness of a child should differ significantly from those of an adult. To account for shape variation, we further condition the model on bone parameter $\boldsymbol{\zeta}$.

$$F^{l,\boldsymbol{\zeta}}_{\Theta_\sigma} : (\gamma(\mathbf{x}^l), \gamma(\boldsymbol{\zeta})) \rightarrow (\sigma, \mathbf{h}) \tag{10}$$

Meanwhile, the color c at a local 3D location may change with a transformation of the object coordinate system, as this may lead to changes in the local lighting condition. Since the RGB color c at a local 3D location should further depend on rigid transformation l, we use a 6D vector $\mathfrak{se}(3)$ representation $\boldsymbol{\xi}$ of transformation l as a network input.

$$F^{l,\boldsymbol{\zeta}}_{\Theta_c} : (\mathbf{h}, \gamma(\mathbf{d}^l), \gamma(\boldsymbol{\xi})) \rightarrow (c) \tag{11}$$

where $\mathbf{d}^l = R^{-1}\mathbf{d}$ is the 2D view direction in the object coordinate system.

Combining Eqs. 10-11, the rigidly transformed neural radiance field (RT-NeRF) defined in the $l = \{R, \mathbf{t}\}$ space is expressed as

$$F^{l,\boldsymbol{\zeta}}_{\Theta} : (\gamma(\mathbf{x}^l), \gamma(\mathbf{d}^l), \gamma(\boldsymbol{\xi}), \gamma(\boldsymbol{\zeta})) \rightarrow (c, \sigma) \tag{12}$$

RT-NeRF serves as the basic building block in the neural articulated radiance field, and we will next show how it is utilized to overcome the 'Implicit Transformations' and 'Part Dependency' issues.

3.4. Neural Articulated Radiance Field

The proposed neural articulated radiance field (NARF) is built upon RT-NeRF. We first introduce two basic solutions, *Part-Wise NARF* and *Holistic NARF*, and analyze the

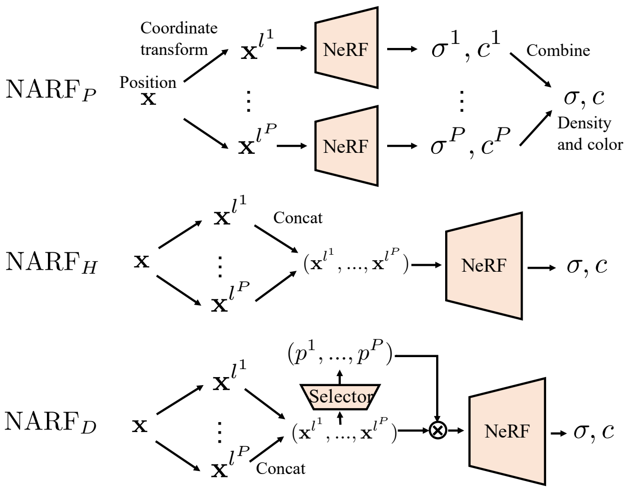

Figure 2. Three types of neural articulated radiance field. Inputs other than position $\mathbf{x}$ are omitted for better understanding.

pros and cons of each. Then, we propose our final solution named *Disentangled NARF* that shares the merits of both *Part-Wise* and *Holistic* NARF. Conceptual figures are visualized in Figure 2.

Part-Wise NARF (NARF$_P$) Given a kinematic 3D pose configuration of an articulated object $\{\mathbf{p}, \mathbf{o}, \boldsymbol{\zeta}, \theta\}$, we first compute the global rigid transformation $\{l^i | i = 1, ..., P\}$ for each rigid part using the forward kinematics in Eqs. 6-7. To estimate the density and color (σ, c) of a global 3D location $\mathbf{x}$ from a 2D viewing direction $\mathbf{d}$, we train a separate RT-NeRF, $F^{l^i, \boldsymbol{\zeta}}_{\Theta^i}$,

$$\mathbf{x}^{l^i} = R^{i^{-1}}(\mathbf{x} - \mathbf{t}^i), \mathbf{d}^{l^i} = R^{i^{-1}}\mathbf{d} \tag{13}$$

$$F^{l^i, \boldsymbol{\zeta}}_{\Theta^i} : (\gamma(x^{l^i}), \gamma(\mathbf{d}^{l^i}), \gamma(\boldsymbol{\xi}^i), \gamma(\boldsymbol{\zeta})) \rightarrow (c^i, \sigma^i), \tag{14}$$

for each part using Eq. 12 and combine the densities and colors $\{\sigma^i, c^i | i = 1, ..., P\}$ estimated by different RT-NeRFs into one. We denote this approach as ***Part-Wise NARF*** (NARF$_P$).

Since a particle on an object can belong to only one of the object parts, only one of the estimates in $\{\sigma^i, c^i | i = 1, ..., P\}$ should be nonzero. The density and color (σ, c) of a global 3D location $\mathbf{x}$ can be determined by taking the estimate with the highest density. However, the max operation is not differentiable, so we instead use the softmax function, which is a differentiable weighted sum over all the estimates:

$$\sigma = \frac{\sum_{i=1}^{P} \exp(\sigma^i/\tau)\sigma^i}{\sum_{i=1}^{P} \exp(\sigma^i/\tau)}, c = \frac{\sum_{i=1}^{P} \exp(\sigma^i/\tau)c^i}{\sum_{i=1}^{P} \exp(\sigma^i/\tau)}, \tag{15}$$

where τ is the temperature parameter of the softmax function. Volume rendering is then applied using the combined density σ and color c to generate the rendered color $\mathbf{C}(\mathbf{r})$ by Eq. 5. Since the rendering and softmax operations are both differentiable, the image reconstruction loss can pass gradients to all the RT-NeRF models for effective training.

We note that, in addition to color $\mathbf{C}(\mathbf{r})$, the foreground mask $\mathbf{M}(\mathbf{r})$ can also be estimated as an integral of the opacity along the camera ray:

$$\mathbf{M}(\mathbf{r}) = \sum_{j=1}^{N} T_j(1 - \exp(-\sigma_j \delta_j)) \tag{16}$$

We can further render a segmentation image that indicates which RT-NeRF (object part) is used for rendering each pixel:

$$s_j(\mathbf{r}) = \arg\max_i \{\sigma^i_j | i \in [1, P]\} \tag{17}$$

$$\mathbf{S}^i(\mathbf{r}) = \sum_{j=1}^{N} T_j(1 - \exp(-\sigma_j \delta_j))(s_j(\mathbf{r}) == i) \tag{18}$$

where s_j is the index of the part with the greatest density, and $\mathbf{S}^i$ denotes a segmentation mask for the i^{th} part.

Discussion. The NARF$_P$ approach models the rigidly transformed parts of an articulated object in separate RT-NeRFs, where each part has a consistent radiance field under different 3D pose configurations. As the rigid transformation of each RT-NeRF is computed explicitly via forward kinematics, rather than implicitly within the network, the issue of implicit transformations and part dependency are avoided. The part dependency issue is also addressed by taking the estimate with the highest density while suppressing the contribution of other parts for a global 3D location in Eq. 15. However, its computation is inefficient for the following reasons.

- The computational cost is proportional to the number of object parts, limiting the representation capacity for complex articulated objects.
- Training is dominated by the large number of zero density point samples. As a particle on an object can belong to only one of the object parts, it will be trained as a zero density sample for the remaining parts. Since parts with small densities do not affect the value of the equation very much, it is not really necessary to calculate the density of those parts.

Holistic NARF (NARF$_H$) To address the above issues, we present another approach that combines the inputs of the RT-NeRF models in NARF$_P$ then feeds them as a whole into a single NeRF model for direct regression of the final density and color (σ, c). We call this approach ***Holistic NARF*** (NARF$_H$). Formally,

$$F^{l, \boldsymbol{\zeta}}_{\Theta_\sigma} : \text{Cat}(\{\gamma(\mathbf{x}^{l^i}) | i \in [1, P]\}, \gamma(\boldsymbol{\zeta})) \rightarrow (\sigma, \mathbf{h}), \tag{19}$$

$$F^{l,\boldsymbol{\zeta}}_{\Theta_c}: \mathrm{Cat}(\mathbf{h}, \{(\gamma(\mathbf{d}^{l^i}), \gamma(\boldsymbol{\xi}^i))|i \in [1, P]\}) \rightarrow (c), \quad (20)$$

where Cat denotes the concatenation operator.

Discussion. There is only a single NeRF model trained in NARF$_H$. The computational cost is almost constant to the number of object parts and the zero density problem is naturally avoided. However, unlike ***Part-Wise NARF***, NARF$_H$ does not satisfy ***Part Dependency***, because all parameters are considered for each 3D location. Moreover, object part segmentation masks cannot be generated from Eq. 18 without part dependencies.

Disentangled NARF (NARF$_D$) We propose ***Disentangled NARF*** (NARF$_D$) which shares the merits of both NARF$_P$ and NARF$_H$ while avoiding their weaknesses by introducing a selector $\mathcal{S}$.

The selector $\mathcal{S}$ identifies which object part a global 3D location $\mathbf{x}$ belongs to. $\mathcal{S}$ consists of P lightweight sub-networks for each part. For the i^{th} part, a sub-network O^i_Γ takes the local 3D position of $\mathbf{x}$ in $l^i = \{R^i, \mathbf{t}^i\}$ and the bone parameter $\boldsymbol{\zeta}$ as input and outputs the probability p^i of $\mathbf{x}$ belonging to the i^{th} part. Since $\mathbf{x}$ should be assigned to only one of the object parts, the softmax activation is used to normalize the selector's outputs:

$$O^i_\Gamma: (\gamma(\mathbf{x}^{l^i}), \gamma(\boldsymbol{\zeta})) \rightarrow (o^i), \quad p^i = \frac{\exp(o^i)}{\sum_{k=1}^{P}\exp(o^k)} \quad (21)$$

It can be seen that O^i_Γ is actually an occupancy network [39, 8] defined *in the local object coordinate system*. For implementation, we use a two-layer MLP with ten hidden nodes for each occupancy net, which is lightweight yet effective.

Disentangled NARF is defined by (softly) masking out the irrelevant parts in the concatenated input using the outputs p^i of the selector.

$$F^{l,\boldsymbol{\zeta}}_{\Theta_\sigma}: \mathrm{Cat}(\{\gamma(\mathbf{x}^{l^i}) * p^i|i \in [1, P]\}, \gamma(\boldsymbol{\zeta})) \rightarrow (\sigma, \mathbf{h}), \quad (22)$$

$$F^{l,\boldsymbol{\zeta}}_{\Theta_c}: \mathrm{Cat}(\mathbf{h}, \{(\gamma(\mathbf{d}^{l^i}) * p^i, \gamma(\boldsymbol{\xi}^i) * p^i)|i \in [1, P]\}) \rightarrow (c), \quad (23)$$

Note that though we have removed the dependency on irrelevant parts by masking their inputs, the resulting input is still in the form of a concatenation. This is done purposely because all the bones share a single RT-NeRF, which needs to distinguish the different bones in order to generate the corresponding density and color. Different bones are distinguished by dimensions of the concatenated vector that represent them, similar to part identity encoding.

Since the selector outputs the probabilities of a global 3D location belonging to each part, we can generate the segmentation mask by selecting the locations occupied by a specific part followed by Eq. 18:

$$s_j(\mathbf{r}) = \arg\max_i \{p^i_j | i \in [1, P]\} \quad (24)$$

3.5. Training Details

During training, at each optimization iteration, we randomly sample a batch of camera rays from the set of all pixels, and then follow the hierarchical volume sampling strategy of the original NeRF [41] to query N samples for each ray. With known kinematic 3D pose configuration $\{\mathbf{p}, \mathbf{o}, \boldsymbol{\zeta}, \theta\}$, the samples' densities and colors are estimated by the NARF model. Volume rendering is then used to render the color $\mathbf{C}(\mathbf{r})$ and mask $\mathbf{M}(\mathbf{r})$ of this ray using Eq. 5 and 16, respectively. The loss is the total squared error between the rendered and true pixel colors and masks.

$$\mathcal{L} = \sum_{\mathbf{r}\in\mathcal{R}} [||\hat{\mathbf{C}}(\mathbf{r}) - \mathbf{C}(\mathbf{r})||^2_2] + \sum_{\mathbf{r}\in\mathcal{R}} [||\hat{\mathbf{M}}(\mathbf{r}) - \mathbf{M}(\mathbf{r})||^2_2] \quad (25)$$

where $\mathcal{R}$ is the set of rays in each batch, $\hat{\mathbf{C}}$ and $\hat{\mathbf{M}}$ are the ground truth color and foreground mask. In the supplemental material, we empirically show that the extra mask loss helps to learn a cleaner background. Other training details on the learning rate, batch size and optimizer can be found in the supplemental material.

4. Results of Training on a Single Object

In this section, we evaluate our model in the case of a single articulated 3D object.

Dataset and Settings We create our own synthetic dataset of human bodies for experimentation. It consists of two persons, one male and one female, selected from the human 3D textured mesh (THuman) dataset [67]. Each person has 56 and 48 different poses, 26 of which are used for training and the others for testing. We render 100 images with various orientations and scaling for each mesh for training and 20 for testing, ending up with 2600 training images for each person. We denote this test data setting as the **novel pose/novel view** setting. All rendered images have a resolution of 128 × 128. Additionally, we introduce three other test settings for a more comprehensive comparison. The **same pose/same view** setting uses testing images rendered from the same poses and same viewpoint distribution as in training. The **novel pose/same view** setting uses novel poses but the viewpoint distribution is the same as in training. Finally, the **same pose/novel view** setting uses the same poses but the viewpoint distribution is different from training.

Metrics Three metrics are used to evaluate performance. The peak signal to noise ratio (PSNR) and structural similarity index (SSIM) [60] are two commonly used evaluation metrics for image reconstruction (higher is better). In addition, we introduce the L2 distance error of mask images

Method	Cost			Same pose, same view			Novel pose, same view			Same pose, novel view			Novel pose, novel view		
	#Params	#FLOPS	#Memory	Mask↓	PSNR ↑	SSIM ↑	Mask ↓	PSNR ↑	SSIM ↑	Mask ↓	PSNR ↑	SSIM ↑	Mask ↓	PSNR ↑	SSIM ↑
CNN	15.6M	-	-	4.694	29.12	0.9429	8.231	27.30	0.9211	22.33	25.19	0.8532	23.93	24.53	0.8470
P-NeRF	0.85M	156M	356K	47.52	21.42	0.8006	65.73	20.42	0.7696	51.56	21.19	0.7897	67.75	20.27	0.7648
D-NARF	0.66M	121M	382K	133.2	18.90	0.1143	140.8	18.81	0.1140	130.4	19.09	0.1144	136.7	18.88	0.1133
NARF$_P$	11.8M	2140M	6544K	5.612	28.56	0.9258	7.095	26.83	0.9052	6.195	27.54	0.9144	7.681	26.50	0.9104
NARF$_H$	1.06M	197M	344K	3.394	29.91	0.9470	22.99	24.09	0.8665	4.303	28.81	0.9370	22.86	23.98	0.8646
NARF$_D$	1.10M	205M	382K	**3.080**	**30.86**	**0.9586**	**6.980**	**27.93**	**0.9317**	**3.914**	**29.44**	**0.9466**	**7.554**	**27.24**	**0.9230**

Table 1. Quantitative comparison for a single object. Best results in **bold**.

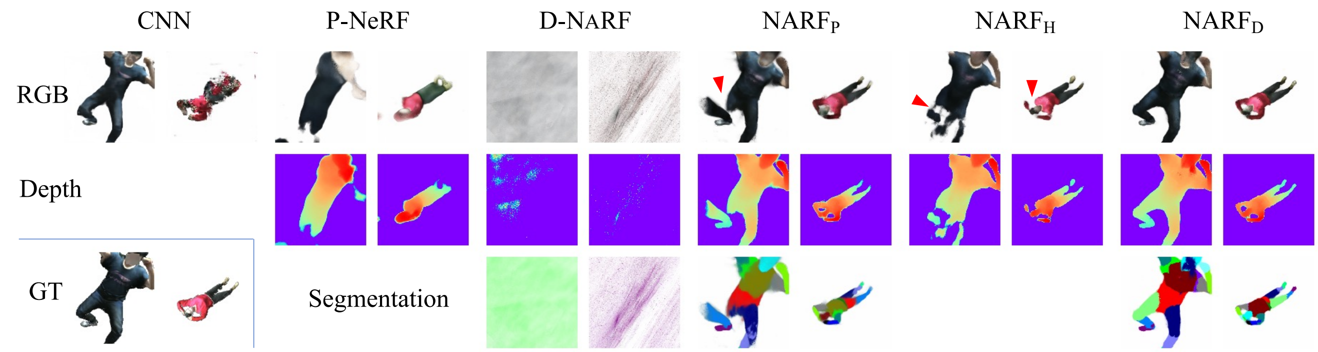

Figure 3. Generative results comparison for a single object with a novel pose and novel view. One model per person identity. Triangles point to areas that should be noted. NARF$_D$ best generalizes to novel view/novel poses among the six methods.

Figure 4. Disentangled representations learned by NARF$_D$. For bone parameter interpolation, the right leg length is interpolated.

(Mask), which better describes how close the 3D shape of the rendered object is to the ground truth (lower is better).

Baselines In addition to the three variants of NARF (NARF$_P$, NARF$_H$ and NARF$_D$), three other baselines are included for comparison. The first is a 2D ***CNN-based*** method similar to [4] that generates the target subject image from "pose stick figures". The pose stick figures in our case are generated by projecting the 3D joints into the 2D image (with given camera parameters) then adding lines to connect these 2D keypoints. The second is the ***P-NeRF*** method described in Sec. 3.2. The third one is ***D-NARF***, a simple extension of D-NERF [45] to articulated objects. D-NARF aims to learn the mapping $\Psi : \mathbf{x} \rightarrow \mathbf{x}'$ that transforms a given point to its position in a canonical shape space. In our implementation, a static NeRF model for a canonical pose $\mathcal{P}^c$ is learned, then a mapping network [45] Ψ estimates the deformation field between the scene of a specific pose instance $\mathcal{P}$ and the scene of the canonical pose $\mathcal{P}^c$. The details of the three baselines can be found in the supplemental material.

Results Quantitative comparison results are given in Table 1. #Params, #FLOPS, and #Memory denote the number of parameters, floating point operations per ray, and number of elements to preserve during forward propagation per ray, which is proportional to the memory cost.

It can be seen that our method, **NARF$_D$**, outperforms the others under all the evaluation metrics and test data settings (best results shown in **bold**). Particularly, it exhibits high performance under novel pose and/or novel view settings (slight performance drop on SSIM within 4%) with low computational cost (close to a single NeRF model, P-NeRF). Hence, we can conclude that NARF$_D$ effectively and efficiently learns the radiance field of an articulated 3D object and the model generalizes to novel poses and views with high fidelity.

In contrast, all the other methods are deficient in one way or another. The **CNN-based** method [4] fails when tested under novel views (9.68% performance drop on SSIM) since it is difficult to learn an effective 3D representation from 2D inputs. **P-NeRF** and **D-NARF** fail in almost all cases mainly due to both the Implicit Transformations and Part Dependency issues. **NARF$_P$** exhibits good performance and generalization ability but requires much more computation (10 × #FLOPS and 17 × #Memory of **NARF$_D$**). **NARF$_H$** is less effective when tested on novel poses (7.85% performance drop on SSIM) due to the Part Dependency issue.

Qualitative results under the **novel pose/novel view** setting are shown in Figure 3. Rendered RGB images (first row), depth maps (second row), and part segmentation maps (third row), as well as ground truth RGB images (bottom left corner) are displayed. It can be seen that the NeRF based methods (except for the CNN-based one) can obtain depth images and the "part dependent" methods (NARF$_P$ and NARF$_D$) can obtain segmentation maps. Our final solution NARF$_D$ generates higher quality RGB, depth and segmentation maps for novel views and poses than the others. Moreover, as shown in Figure 4, NARF$_D$ learns a disentangled representation of camera viewpoint, bone parameters and pose, allowing these appearance properties to be individually controlled in rendering.

5. Appearance Variation with Autoencoder

In this section, we train an autoencoder based on NARF to model appearance variation among multiple articulated objects. The autoencoder consists of an encoder and decoder. First, a 2D CNN-based encoder is used to generate a latent vector $\mathbf{z}$ from an input image. The obtained latent vector together with the given camera viewpoint and human pose are fed into our NARF based decoder to reconstruct the input image.

Following the implementation for the NeRF-based generator [52], we first decompose $\mathbf{z}$ into a shape latent vector $\mathbf{z}_s$ and an appearance latent vector $\mathbf{z}_a$. Then, $\mathbf{z}_s$ is concatenated to the density-dependent inputs, namely, the positionally encoded location $\mathbf{x}$ and bone parameters $\boldsymbol{\zeta}$. Meanwhile, $\mathbf{z}_a$ is concatenated to the color-dependent inputs, namely, the positionally encoded view direction $\mathbf{d}$ and the local transformation $\boldsymbol{\xi}$. Specifically, when combining the autoencoder with the NARF$_D$ model, we have

$$F_{\Theta_\sigma}^{l,\boldsymbol{\zeta},\mathbf{z}} : \text{Cat}(\{\gamma(\mathbf{x}^{l^i}) * p^i | i \in [1, P]\}, \gamma(\boldsymbol{\zeta}), \mathbf{z}_s) \rightarrow (\sigma, \mathbf{h}), \tag{26}$$

$$F_{\Theta_c}^{l,\boldsymbol{\zeta},\mathbf{z}} : \text{Cat}(\mathbf{h}, \{(\gamma(\mathbf{d}^{l^i}) * p^i, \gamma(\boldsymbol{\xi}^i) * p^i) | i \in [1, P]\}, \mathbf{z}_a) \rightarrow (c). \tag{27}$$

The encoder and decoder are trained jointly using the same loss in Eq. 25. For the experiment, we use the best performing NARF$_D$ by default. Comparison results for other models can be found in the supplemental material.

Dataset We create another synthetic human body dataset from THuman [67] for experimentation. All of the males (112 in total) and poses (35 on average for each person) in the THuman dataset are used. We render 10 images for each pose with randomly sampled viewpoints to generate 35450 images for training and 3940 images for evaluation. All rendered images have a resolution of 128 × 128.

Results Figure 5 shows the reconstructed RGB images as well as the additional depth images and segmentation maps generated from the input RGB images. A single autoencoder is trained for all objects, indicating that appearance variation is effectively modeled. Figure 6 shows that the NARF-based autoencoder learns a disentangled representation of camera viewpoint, bone parameters, human pose, and color appearance, allowing these properties to be individually controlled in rendering. For color appearance, it is controlled by replacing the appearance latent vector $\mathbf{z}_a$ with that of another person. Additional results can be found in the supplemental material.

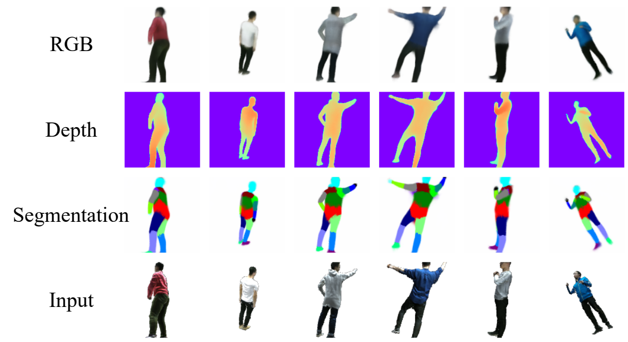

Figure 5. Single-view reconstruction results from NARF$_D$ with an autoencoder. All outputs are generated from a single model.

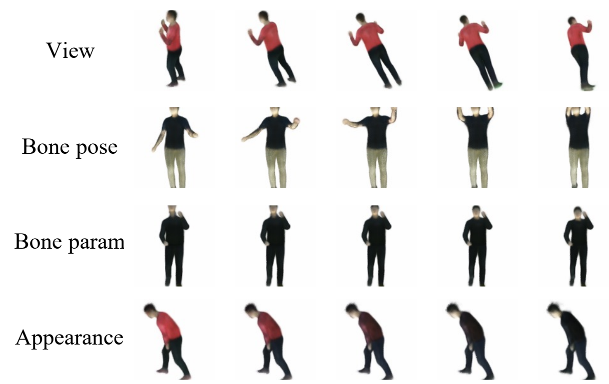

Figure 6. Disentangled representations learned with an autoencoder. For bone parameter interpolation, the head position is interpolated.

6. Conclusion and Future work

In this paper, we propose a method for learning implicit representations for articulated objects. We show that it is possible to learn explicitly controllable representations of viewpoint, pose, bone parameters, and appearance from 3D pose annotated images. Although pose annotation is required to train the model, the model is differentiable and thus may be extended to reduce the required supervisory information, for example, by simultaneously training 3D pose estimation and segmentation with the model. In addition, since the proposed representation provides explicit 3D shape and part segmentation, it may be applied to unsupervised depth estimation and segmentation learning.

7. Acknowledgement

This work was supported by D-CORE Grant from Microsoft Research Asia and partially supported by JST AIP Acceleration Research Grant Number JPMJCR20U3, and JSPS KAKENHI Grant Number JP19H01115. We would like to thank Sho Maeoki, Thomas Westfechtel, and Yang Li for the helpful discussions. We also thank for the support of the GPU resources provided by Microsoft Azure Machine Learning.

References

[1] Guha Balakrishnan, Amy Zhao, Adrian V Dalca, Fredo Durand, and John Guttag. Synthesizing images of humans in unseen poses. In *CVPR*. 2

[2] Oene Bottema and Bernard Roth. *Theoretical kinematics*, volume 24. 1990. 2, 3

[3] Arunkumar Byravan and Dieter Fox. Se3-nets: Learning rigid body motion using deep neural networks. In *ICRA*, 2017. 1

[4] Caroline Chan, Shiry Ginosar, Tinghui Zhou, and Alexei A Efros. Everybody dance now. In *ICCV*, 2019. 1, 2, 7

[5] Eric Chan, Marco Monteiro, Petr Kellnhofer, Jiajun Wu, and Gordon Wetzstein. pi-gan: Periodic implicit generative adversarial networks for 3d-aware image synthesis. In *arXiv*, 2020. 3

[6] Zhiqin Chen and Hao Zhang. Learning implicit fields for generative shape modeling. In *CVPR*. 2

[7] Paul E Debevec, Camillo J Taylor, and Jitendra Malik. Modeling and rendering architecture from photographs: A hybrid geometry-and image-based approach. In *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, 1996. 1, 2

[8] Boyang Deng, JP Lewis, Timothy Jeruzalski, Gerard Pons-Moll, Geoffrey Hinton, Mohammad Norouzi, and Andrea Tagliasacchi. Nasa: Neural articulated shape approximation. In *ECCV*, 2020. 2, 3, 6

[9] Alexey Dosovitskiy, Jost Tobias Springenberg, and Thomas Brox. Learning to generate chairs with convolutional neural networks. In *CVPR*, 2015. 1

[10] Patrick Esser, Ekaterina Sutter, and Björn Ommer. A variational u-net for conditional appearance and shape generation. In *CVPR*. 2

[11] 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 *CVPR*, 2019. 1

[12] Kyle Genova, Forrester Cole, Avneesh Sud, Aaron Sarna, and Thomas Funkhouser. Local deep implicit functions for 3d shape. In *CVPR*, 2020. 1

[13] Kyle Genova, Forrester Cole, Avneesh Sud, Aaron Sarna, and Thomas Funkhouser. Local deep implicit functions for 3d shape. In *CVPR*, 2020. 3

[14] Kyle Genova, Forrester Cole, Daniel Vlasic, Aaron Sarna, William T Freeman, and Thomas Funkhouser. Learning shape templates with structured implicit functions. In *CVPR*, 2019. 3

[15] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *NIPS*, 2014. 2

[16] Phlipp Henzler, Volker Rasche, Timo Ropinski, and Tobias Ritschel. Single-image tomography: 3d volumes from 2d cranial x-rays. In *Computer Graphics Forum*, volume 37, pages 377–388, 2018. 1

[17] Alec Jacobson, Zhigang Deng, Ladislav Kavan, and JP Lewis. Skinning: Real-time shape deformation. In *ACM SIGGRAPH 2014 Courses*, 2014. 2

[18] Doug L James and Christopher D Twigg. Skinning mesh animations. *ACM Transactions on Graphics (TOG)*, 24(3):399–407, 2005. 2

[19] Chiyu Jiang, Avneesh Sud, Ameesh Makadia, Jingwei Huang, Matthias Nießner, Thomas Funkhouser, et al. Local implicit grid representations for 3d scenes. In *CVPR*, 2020. 1

[20] Justin Johnson, Agrim Gupta, and Li Fei-Fei. Image generation from scene graphs. In *CVPR*, 2018. 1

[21] Hanbyul Joo, Tomas Simon, Xulong Li, Hao Liu, Lei Tan, Lin Gui, Sean Banerjee, Timothy Scott Godisart, Bart Nabbe, Iain Matthews, Takeo Kanade, Shohei Nobuhara, and Yaser Sheikh. Panoptic studio: A massively multiview system for social interaction capture. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2017. 2

[22] James T Kajiya and Brian P Von Herzen. Ray tracing volume densities. *ACM SIGGRAPH computer graphics*, 18(3):165–174, 1984. 2, 3

[23] Abhishek Kar, Christian Häne, and Jitendra Malik. Learning a multi-view stereo machine. In *NIPS*, 2017. 1

[24] Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen. Progressive growing of GANs for improved quality, stability, and variation. In *International Conference on Learning Representations*, 2018. 14

[25] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *ICLR*, 2015. 14

[26] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. In *ICLR*, 2014. 2

[27] Tejas D Kulkarni, Pushmeet Kohli, Joshua B Tenenbaum, and Vikash Mansinghka. Picture: A probabilistic programming language for scene perception. In *Proceedings of the ieee conference on computer vision and pattern recognition*, pages 4390–4399, 2015. 1

[28] Tejas D Kulkarni, Vikash K Mansinghka, Pushmeet Kohli, and Joshua B Tenenbaum. Inverse graphics with probabilistic cad models. *arXiv preprint arXiv:1407.1339*, 2014. 1

[29] Tejas D Kulkarni, Will Whitney, Pushmeet Kohli, and Joshua B Tenenbaum. Deep convolutional inverse graphics network. *arXiv preprint arXiv:1503.03167*, 2015. 1

[30] Christoph Lassner, Gerard Pons-Moll, and Peter V Gehler. A generative model of people in clothing. In *ICCV*. 1, 2

[31] Binh Huy Le and Zhigang Deng. Smooth skinning decomposition with rigid bones. *ACM Transactions on Graphics (TOG)*, 31(6):1–10, 2012. 2

[32] John P Lewis, Matt Cordner, and Nickson Fong. Pose space deformation: a unified approach to shape interpolation and

skeleton-driven deformation. In *Proceedings of the 27th annual conference on Computer graphics and interactive techniques*. 2

[33] Zhengqi Li, Simon Niklaus, Noah Snavely, and Oliver Wang. Neural scene flow fields for space-time view synthesis of dynamic scenes. *arXiv preprint arXiv:2011.13084*, 2020. 3

[34] Lingjie Liu, Weipeng Xu, Michael Zollhoefer, Hyeongwoo Kim, Florian Bernard, Marc Habermann, Wenping Wang, and Christian Theobalt. Neural rendering and reenactment of human actor videos. *ACM Transactions on Graphics (TOG)*, 38(5):1–14, 2019. 1, 2

[35] 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. 2, 15

[36] Liqian Ma, Xu Jia, Qianru Sun, Bernt Schiele, Tinne Tuytelaars, and Luc Van Gool. Pose guided person image generation. In *NIPS*, 2017. 2

[37] Liqian Ma, Qianru Sun, Stamatios Georgoulis, Luc Van Gool, Bernt Schiele, and Mario Fritz. Disentangled person image generation. In *CVPR*. 2

[38] Vikash K Mansinghka, Tejas D Kulkarni, Yura N Perov, and Joshua B Tenenbaum. Approximate bayesian image interpretation using generative probabilistic graphics programs. *arXiv preprint arXiv:1307.0060*, 2013. 1

[39] Lars Mescheder, Michael Oechsle, Michael Niemeyer, Sebastian Nowozin, and Andreas Geiger. Occupancy networks: Learning 3d reconstruction in function space. In *CVPR*, 2019. 1, 2, 3, 6

[40] 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. 1

[41] 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 *ECCV*, 2020. 1, 3, 6

[42] Michael Oechsle, Lars Mescheder, Michael Niemeyer, Thilo Strauss, and Andreas Geiger. Texture fields: Learning texture representations in function space. In *ICCV*, 2019. 3

[43] Jeong Joon Park, Peter Florence, Julian Straub, Richard Newcombe, and Steven Lovegrove. Deepsdf: Learning continuous signed distance functions for shape representation. In *CVPR*, 2019. 1

[44] 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*, 2019. 2, 3

[45] Keunhong Park, Utkarsh Sinha, Jonathan T Barron, Sofien Bouaziz, Dan B Goldman, Steven M Seitz, and Ricardo-Martin Brualla. Deformable neural radiance fields. *arXiv preprint arXiv:2011.12948*, 2020. 3, 7, 14

[46] Sida Peng, Yuanqing Zhang, Yinghao Xu, Qianqian Wang, Qing Shuai, Hujun Bao, and Xiaowei Zhou. Neural body: Implicit neural representations with structured latent codes for novel view synthesis of dynamic humans. In *CVPR*, 2021. 15

[47] Eric Penner and Li Zhang. Soft 3d reconstruction for view synthesis. *ACM Transactions on Graphics (TOG)*, 36(6):1–11, 2017. 1

[48] Albert Pumarola, Enric Corona, Gerard Pons-Moll, and Francesc Moreno-Noguer. D-nerf: Neural radiance fields for dynamic scenes. *arXiv preprint arXiv:2011.13961*, 2020. 3

[49] Randi J Rost, Bill Licea-Kane, Dan Ginsburg, John Kessenich, Barthold Lichtenbelt, Hugh Malan, and Mike Weiblen. *OpenGL shading language*. 2009. 2

[50] Shunsuke Saito, Zeng Huang, Ryota Natsume, Shigeo Morishima, Angjoo Kanazawa, and Hao Li. Pifu: Pixel-aligned implicit function for high-resolution clothed human digitization. In *ICCV*, 2019. 3

[51] Shunsuke Saito, Tomas Simon, Jason Saragih, and Hanbyul Joo. Pifuhd: Multi-level pixel-aligned implicit function for high-resolution 3d human digitization. In *CVPR*, 2020. 3

[52] Katja Schwarz, Yiyi Liao, Michael Niemeyer, and Andreas Geiger. Graf: Generative radiance fields for 3d-aware image synthesis. *NeurIPS*, 2020. 3, 8

[53] Aliaksandr Siarohin, Enver Sangineto, Stéphane Lathuiliere, and Nicu Sebe. Deformable gans for pose-based human image generation. In *CVPR*. 2

[54] Vincent Sitzmann, Julien Martel, Alexander Bergman, David Lindell, and Gordon Wetzstein. Implicit neural representations with periodic activation functions. 2020. 3

[55] Vincent Sitzmann, Michael Zollh ö fer, and Gordon Wetzstein. Scene representation networks: Continuous 3d-structure-aware neural scene representations. 2019. 1, 3

[56] Pratul P Srinivasan, Richard Tucker, Jonathan T Barron, Ravi Ramamoorthi, Ren Ng, and Noah Snavely. Pushing the boundaries of view extrapolation with multiplane images. In *CVPR*, 2019. 1

[57] Maxim Tatarchenko, Alexey Dosovitskiy, and Thomas Brox. Multi-view 3d models from single images with a convolutional network. In *ECCV*. Springer, 2016. 1

[58] Shubham Tulsiani, Tinghui Zhou, Alexei A Efros, and Jitendra Malik. Multi-view supervision for single-view reconstruction via differentiable ray consistency. In *CVPR*, 2017. 1

[59] Ruben Villegas, Jimei Yang, Yuliang Zou, Sungryull Sohn, Xunyu Lin, and Honglak Lee. Learning to generate long-term future via hierarchical prediction. In *ICML*. 2

[60] Zhou Wang, Alan C Bovik, Hamid R Sheikh, and Eero P Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE transactions on image processing*, 13(4), 2004. 6

[61] Jiajun Wu, Joshua B Tenenbaum, and Pushmeet Kohli. Neural scene de-rendering. In *CVPR*, 2017. 1

[62] Jiajun Wu, Tianfan Xue, Joseph J Lim, Yuandong Tian, Joshua B Tenenbaum, Antonio Torralba, and William T Freeman. Single image 3d interpreter network. In *ECCV*, 2016. 1

[63] Xinchen Yan, Jimei Yang, Ersin Yumer, Yijie Guo, and Honglak Lee. Perspective transformer nets: Learning single-view 3d object reconstruction without 3d supervision. In *NIPS*, 2016. 1

[64] Wentao Yuan, Zhaoyang Lv, Tanner Schmidt, and Steven Lovegrove. Star: Self-supervised tracking and reconstruction of rigid objects in motion with neural rendering. *arXiv preprint arXiv:2101.01602*, 2020. 3

[65] Mihai Zanfir, Alin-Ionut Popa, Andrei Zanfir, and Cristian Sminchisescu. Human appearance transfer. In *CVPR*. 1, 2

[66] Ailing Zeng, Xiao Sun, Fuyang Huang, Minhao Liu, Qiang Xu, and Stephen Lin. Srnet: Improving generalization in 3d human pose estimation with a split-and-recombine approach. In *ECCV*, pages 507–523, 2020. 2, 4

[67] Zerong Zheng, Tao Yu, Yixuan Wei, Qionghai Dai, and Yebin Liu. Deephuman: 3d human reconstruction from a single image. In *ICCV*, 2019. 6, 8

[68] Tinghui Zhou, Richard Tucker, John Flynn, Graham Fyffe, and Noah Snavely. Stereo magnification: Learning view synthesis using multiplane images. *arXiv preprint arXiv:1805.09817*, 2018. 1

[69] Tinghui Zhou, Shubham Tulsiani, Weilun Sun, Jitendra Malik, and Alexei A Efros. View synthesis by appearance flow. In *ECCV*, 2016. 1

[70] Xingyi Zhou, Xiao Sun, Wei Zhang, Shuang Liang, and Yichen Wei. Deep kinematic pose regression. In *ECCV*, 2016. 1, 2, 4

Supplementary Materials: Neural Articulated Radiance Field

A. Ablation Studies for AutoEncoder

In Table 1 of the main paper, we quantitatively evaluate our model in the case of a single articulated 3D object, while in Section 5 in the main paper, we only show qualitative results in the case of Autoencoder. Here, we supplement those results with the quantitative ablation study for Autoencoder in Table 2. The same test data settings (namely "same pose/same view", "novel pose/same view", "same pose/novel view", and "novel pose/novel view"), metrics (namely PSNR, SSIM, and Mask), and baseline methods (namely NARF$_P$, NARF$_H$ and NARF$_D$, ***CNN-based***, ***P-NeRF*** and ***D-NARF***) are used for comparison. The same dataset as in Section 5 of the main paper is used for experimentation.

At testing phase, when extracting the latent shape and appearance vectors ($\mathbf{z}_s$ and $\mathbf{z}_a$) using the encoder, we use images under the same viewpoint distribution as in the training images as input. Then, images from novel views and poses are rendered by combining $\mathbf{z}_s$ and $\mathbf{z}_a$ with unseen views and poses in the training data.

Results Quantitative comparison results are given in Table 2. Qualitative results under the novel pose/novel view setting are shown in Figure 7. Consistent with the case of a single object, our method NARF$_D$ outperforms the others under all the evaluation metrics and test data settings (best results shown in **bold**). High quality depth and segmentation images are jointly generated as shown in Figure 7 (rightmost column). CNN based models cannot represent 3D structure effectively, so the performance drops significantly in the "novel view" settings. Figure 7 (left-top) shows that in the novel view testing, the CNN-based method produces fuzzy images. Meanwhile, it cannot generate depth and segmentation images. P-NeRF and D-NARF fail in almost all settings due to implicit transformation and part dependency problems. NARF$_P$ generally performs well, but the computational cost is too high. NARF$_H$ has poor performance on "novel pose" due to part dependency issues. The performance drop on novel pose in the Autoencoder case is not as significant as in the single object case (shown in Table 1 of the main paper) since the pose diversity in the training data is much larger in the Autoencoder case.

B. Ablation Studies in RT-NeRF

In Section 3.3 of the main paper, we introduced the rigidly transformed neural radiance field (RT-NeRF) to effectively model a rigidly transformed object part. Here, we evaluate the effectiveness of the two most critical design elements in RT-NeRF. The first is the ***explicit transformation*** that converts a global 3D location into the *local* coordinate system and the local 3D location is then used to estimate the density using Equations 9 and 10 of the main paper. The second is the ***pose-dependent color*** estimation defined in Equation 11 of the main paper. It takes the 6D vector $\mathfrak{se}(3)$ representation $\boldsymbol{\xi}$ of transformation l as a network input to estimate the RGB color c. To this end, two more baseline methods are introduced accordingly to compare to ***RT-NeRF***. The first is the rigid pose conditioned NeRF (***RP-NeRF***) that takes the global 3D location and the rigid transformation $\boldsymbol{\xi}$ as network inputs, similar to the P-NeRF defined in Equation 8 of the main paper. The second is ***RT-NeRF w/o*** $\boldsymbol{\xi}$ that estimates the RGB color c without using the transformation $\boldsymbol{\xi}$ as input in Equation 11 of the main paper.

Dataset We create a synthetic rigid object dataset of a rendered bulldozer using Blender (a software for rendering) for experimentation. In the dataset, the object (a bulldozer) can rigidly transform in the world coordinate system. For each rendered image, both rigid transformation and the camera viewpoint are randomly set. The camera will be translated to point to the center of the object so that the object will appear in the center of the rendered image. The resolution for all rendered images is set to 200×200. In total, 480 images are used for training and another 20 images are used for testing. The loss function is the same as in Equation 25 of the main paper.

Results The quantitative results are shown in Table 3 and the qualitative results are shown in Figure 8. The experi-

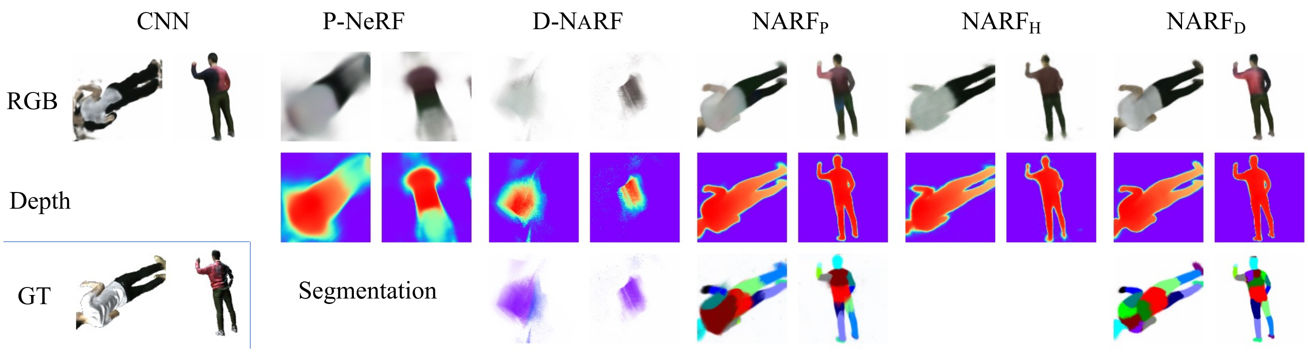

Figure 7. Generative results comparison for AutoEncoder

	Cost			Same pose, same view			Novel pose, same view			Same pose, novel view			Novel pose, novel view		
Method	#Params	#FLPOS	#Memory	Mask↓	PSNR ↑	SSIM ↑	Mask ↓	PSNR ↑	SSIM ↑	Mask ↓	PSNR ↑	SSIM ↑	Mask ↓	PSNR ↑	SSIM ↑
CNN	15.6M	-	-	**0.005**	**25.59**	**0.8966**	**0.010**	**24.70**	**0.8757**	0.023	**22.70**	0.8213	0.023	**22.98**	0.8243
P-NeRF	0.85M	156M	356K	0.093	19.24	0.5362	0.096	19.60	0.5346	0.117	18.13	0.4666	0.120	19.08	0.4746
D-NARF	0.66M	121M	382K	0.126	18.27	0.6733	0.126	18.81	0.6706	0.124	18.00	0.5983	0.125	18.82	0.6121
NARF_P	11.8M	<u>2140M</u>	<u>6544K</u>	0.010	22.57	0.8250	0.011	22.58	0.8211	0.012	21.32	0.8056	0.013	21.80	0.8088
NARF_H	1.06M	197M	344K	0.012	22.89	0.8244	0.014	23.14	0.8205	0.016	21.48	0.7891	0.017	22.36	0.7961
NARF_D	1.10M	205M	382K	0.008	23.84	0.8568	**0.010**	23.55	0.8435	**0.010**	22.25	**0.8313**	**0.011**	22.81	**0.8294**

Table 2. Quantitative comparison for autoencoders. Best results in **bold**.

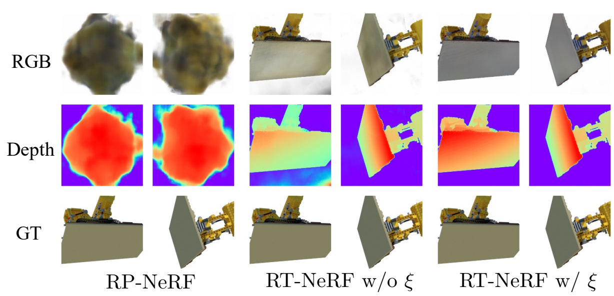

Figure 8. Qualitative results for RT-NeRF comparison

Method	Mask↓	PSNR ↑	SSIM ↑
RP-NeRF	275.331	11.25	0.3103
RT-NeRF w/o ξ	1.783	19.83	0.8255
RT-NeRF	**1.032**	**20.05**	**0.8388**

Table 3. RT-NeRF quantitative comparison. Best results in **bold**.

mental results show that RP-NeRF is unable to learn a good 3D representation and fails to generalize to novel poses and views. In contrast, RT-NeRF effectively models the rigidly transformed object by *explicitly* transforming the global 3D location into the local coordinate system. In addition, the color estimation without the transformation input is less effective. This is concluded by comparing the results of "RT-NeRF w/o ξ" with the results of "RT-NeRF". Quantitatively, the performance of "RT-NeRF w/o ξ" drops significantly under the Mask metric in Table 3. Qualitatively, the rendered images from "RT-NeRF w/o ξ" look blurry compared to "RT-NeRF" in Figure 8.

C. Additional Ablation Studies in NARF

In this section, we provide more ablation studies on the effects of the mask loss (the second item in Equation 25 of the main paper), temperature parameter for NARF_P (in Equation 15 of the main paper), and softmax activation function for NARF_D (in Equation 21 of the main paper).

W/ and w/o mask loss We conduct experiments to evaluate the effect of the mask loss (the second term of Equation 25 of the main paper) added to the rendered mask image. While the color loss optimizes the final RGB color of the rendered pixels, the gradient from the mask loss directly optimizes the densities of the particles. Therefore, the additional mask loss is helpful in learning 3D shapes efficiently. The quantitative and qualitative results of w/ and w/o mask loss are shown in Table 4 and Figure 9 respectively. In Table 4, it can be seen that performances of all the three variants of NARF drop significantly without the mask loss, especially under the novel pose/novel view setting. Particularly, in Figure 9, the NARF_H model is not able to converge at all without the mask loss. The NARF_P and NARF_D models still work without the mask loss but the rendered images get very blurry, especially on the background regions around the object.

	Cost			Same pose, same view			Novel pose, same view			Same pose, novel view			Novel pose, novel view		
Method	#Params	#FLPOS	#Memory	Mask L2 ↓	PSNR ↑	SSIM ↑	Mask L2 ↓	PSNR ↑	SSIM ↑	Mask L2 ↓	PSNR ↑	SSIM ↑	Mask L2 ↓	PSNR ↑	SSIM ↑
CNN	15.6M	-	-	4.694	29.12	0.9429	8.231	27.30	0.9211	22.33	25.19	0.8532	23.93	24.53	0.8470
P-NeRF	0.85M	156M	356K	47.52	21.42	0.8006	65.73	20.42	0.7696	51.56	21.19	0.7897	67.75	20.27	0.7648
D-NARF	0.66M	121M	382K	133.2	18.90	0.1143	140.8	18.81	0.1140	130.4	19.09	0.1144	136.7	18.88	0.1133
NARF$_P$	11.8M	<u>2149M</u>	<u>6544K</u>	5.612	28.56	0.9258	7.095	26.83	0.9052	6.195	27.54	0.9144	7.681	26.50	0.9104
NARF$_H$	1.06M	197M	344K	3.394	29.91	0.9470	22.99	24.09	0.8665	4.303	28.81	0.9370	22.86	23.98	0.8646
NARF$_D$	1.10M	205M	382K	**3.080**	**30.86**	**0.9586**	**6.980**	**27.93**	**0.9317**	**3.914**	**29.44**	**0.9466**	**7.554**	**27.24**	**0.9230**
NARF$_D sigmoid$	1.10M	205M	382K	3.103	30.74	0.9578	7.181	27.83	0.9304	3.942	29.35	0.9459	7.655	26.17	0.9129
NARF$_P$32	0.32M	59M	824K	7.009	27.85	0.9191	8.539	26.86	0.9068	7.503	27.14	0.9105	8.988	25.74	0.8933
NARF$_P$64	0.98M	178M	1641K	6.442	27.86	0.9212	7.747	26.78	0.9094	6.944	27.21	0.9127	8.122	26.28	0.9041
NARF$_P$128	3.28M	596M	3275K	5.722	28.46	0.9284	7.088	27.15	0.9135	6.281	27.56	0.9180	7.582	26.23	0.9068
NARF$_P$256	11.8M	2149M	6544K	5.612	28.56	0.9258	7.095	26.83	0.9052	6.195	27.54	0.9144	7.681	26.50	0.9104
NARF$_P, \tau = 0.01$	0.32M	59M	824K	14.656	25.81	0.8730	17.212	24.92	0.8541	15.623	25.32	0.8624	17.569	23.72	0.8333
NARF$_P, \tau = 0.1$	0.32M	59M	824K	10.259	26.79	0.8995	12.006	25.92	0.8863	10.747	26.26	0.8911	12.502	25.10	0.8756
NARF$_P, \tau = 1$	0.32M	59M	824K	8.639	27.24	0.9077	10.372	26.29	0.8949	9.208	26.56	0.8989	10.876	25.16	0.8809
NARF$_P, \tau = 10$	0.32M	59M	824K	7.009	27.85	0.9191	8.539	26.86	0.9068	7.503	27.14	0.9105	8.988	25.74	0.8933
NARF$_P, \tau = 100$	0.32M	59M	824K	6.585	27.83	0.9190	7.944	26.96	0.9059	7.077	27.23	0.9117	8.393	26.24	0.8983
NARF$_P, \tau = 1000$	0.32M	59M	824K	6.651	27.44	0.9137	7.985	26.68	0.9014	7.138	26.91	0.9072	8.443	26.08	0.8953
NARF$_P$ w/o L_{mask}	0.32M	59M	824K	11.834	28.01	0.9131	13.442	26.72	0.8994	12.738	27.33	0.9028	14.026	25.54	0.8792
NARF$_H$ w/o L_{mask}	1.06M	197M	344K	416.026	24.25	0.7004	420.720	21.10	0.6636	419.693	23.40	0.7029	424.636	20.53	0.5244
NARF$_D$ w/o L_{mask}	1.10M	205M	382K	6.922	**31.39**	**0.9630**	10.116	**28.09**	**0.9346**	8.107	**29.90**	**0.9508**	16.098	25.71	0.8889

Table 4. Quantitative results of ablation studies.

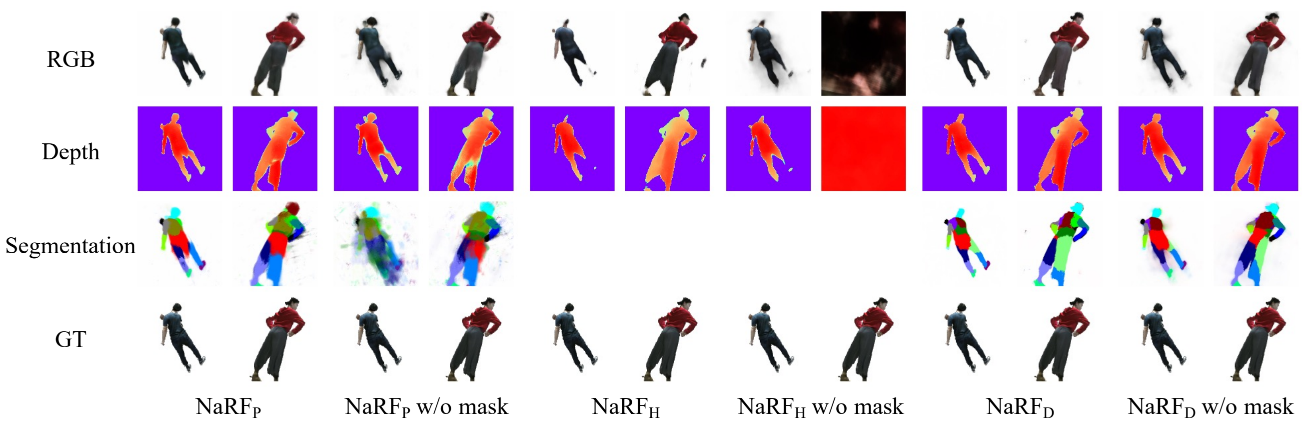

Figure 9. Comparison of mask loss.

Temperature parameter τ in NARF$_P$ We study the effect of the temperature parameter $\tau \in (0, \infty)$ in NARF$_P$ (in Equation 15 of the main paper). The temperature parameter determines how soft the selection is among the multiple RT-NeRFs. When τ is close to 0, hard selection is performed. Though the ***Part Dependency*** prior is strictly satisfied in this case, convergence in the training is difficult since the highest current estimate will completely block the gradient from back-propagating to the others. It is especially worse in the early stage of the training when the highest estimate is almost random. In turn, when τ is close to ∞, averaging is performed. In this case, the gradient is back-propagated to all RT-NeRFs, but the ***Part Dependency*** issue arises again, which will harm the generalization ability to novel poses. The quantitative and qualitative results are shown in Table 4 and Figure 10, respectively. We empirically use the best-performing $\tau = 100$ setting as shown in Table 4.

Softmax vs. sigmoid activation in NARF$_D$ In Equation 21 of the main paper, we use softmax activation, which is also motivated by the ***Part Dependency*** prior. Here, we provide the results of using the sigmoid activation as an alternative. Formally, Equation 21 of the main paper is replaced with Equation 28:

$$O^i_\Gamma : (\gamma(\mathbf{x}^{l^i}), \gamma(\boldsymbol{\zeta})) \rightarrow (o^i), \quad p^i = \frac{1}{1 + \exp(-o^i)} \tag{28}$$

The quantitative results are shown in Table 4. In conclusion, softmax outperforms sigmoid, especially in the "novel pose/novel view" setting.

D. Implementation Details in P-NeRF

In Equation 8 of the main paper, P-NeRF takes a global 3D location $\mathbf{x}$ and the part transformations $\{l^i | i = 1, ..., P\}$ as input. For implementation, we use the 6D vector $\mathfrak{se}(3)$ representation $\boldsymbol{\xi}^i$ of transformation l^i, and concatenate them together with $\mathbf{x}$ and the bone length ζ as the network input.

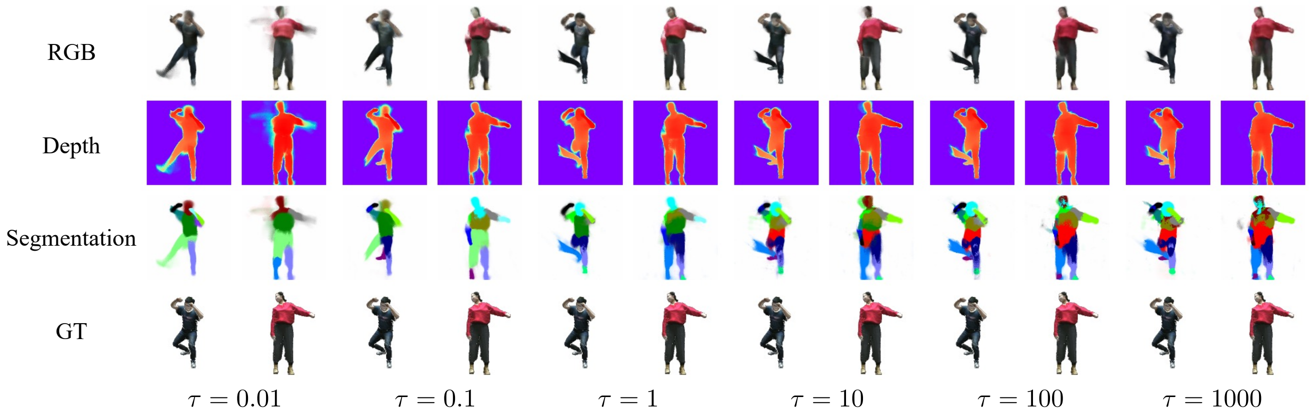

Figure 10. Comparison of temperature τ of NARF$_P$.

Positional encoding is performed before the concatenation. Formally,

$$F_{\Theta}^{\mathcal{P}} : (\gamma(\mathbf{x}), \{\gamma(\xi^i) | i = 1, ..., P\}, \gamma(\zeta), \gamma(\mathbf{d})) \to (\sigma, c). \quad (29)$$

The density and color sub-networks are defined as

$$F_{\Theta_\sigma}^{\mathcal{P}} : (\gamma(\mathbf{x}), \{\gamma(\xi^i) | i = 1, ..., P\}, \gamma(\zeta)) \to (\sigma, \mathbf{h}), \quad (30)$$
$$F_{\Theta_c}^{\mathcal{P}} : (\mathbf{h}, \{\gamma(\xi^i) | i = 1, ..., P\}, \gamma(\mathbf{d})) \to (c). \quad (31)$$

E. Implementation Details in D-NARF

D-NERF [45] uses a canonical template and learns the observation-to-canonical deformation

$$\Psi : (\mathbf{x}, \omega) \to \mathbf{x}' \quad (32)$$

where ω is a deformation latent code.

D-NERF is defined on the deformed position $\mathbf{x}'$ in the canonical template,

$$G : (\Psi(\mathbf{x}, \omega), \mathbf{d}, \psi) \to (\sigma, c) \quad (33)$$

where ψ is a latent appearance code.

In our case, ω and ψ correspond to the pose configuration $\mathcal{P}$ and the appearance latent vector $\mathbf{z}_a$ respectively. Ψ is implemented using MLP, which seems to suffer from the problem of implicit transformation. In our setup, the pose of each part is given, so it can be implemented more directly. In our implementation, we first use the occupancy network similar to the one defined in Equation 21 of the main paper to decide which part the input point belongs to.

$$O_{\Gamma}^i : (\gamma(\mathbf{x}^{l^i}), \gamma(\boldsymbol{\zeta})) \to (o^i), \quad (34)$$
$$p^i = \frac{\exp(o^i)}{\sum_{k=1}^{P} \exp(o^k)}, \quad (35)$$

Then, we calculate the coordinates $\mathbf{x}'$ on the canonical shape as

$$\mathbf{x}' = \sum_i (\mathbf{x}^{l^i} + \mathbf{t}_{\text{canonical}}^i) * p_i \quad (36)$$

where $\mathbf{t}_{\text{canonical}}^i$ is the origin of a canonical pose's i^{th} part in the global coordinate system. View direction in the canonical space and the transformation vector are defined as

$$\mathbf{d} = \sum_i \mathbf{d}^{l^i} * p_i, \ \boldsymbol{\xi} = \sum_i \boldsymbol{\xi}^i * p_i \quad (37)$$

Then, the D-NARF we have implemented in the experiment is defined as

$$F_{\Theta_\sigma}^{l,\boldsymbol{\zeta}} : (\gamma(\mathbf{x}'), \gamma(\boldsymbol{\zeta})) \to (\sigma, \mathbf{h}), \quad (38)$$

$$F_{\Theta_c}^{l,\boldsymbol{\zeta}} : (\mathbf{h}, \gamma(\mathbf{d}), \gamma(\boldsymbol{\xi})) \to (c). \quad (39)$$

This implementation is similar to the implementation of NARF$_D$, differing only in how the coordinates are input to the model. The results of the experiments show that a concatenation-based NARF$_D$ model that retains the coordinates for all parts is more effective than a transformation on the input coordinates in D-NARF.

F. Training Details

We used the Adam [25] optimizer with an initial equalized learning rate [24] of 0.01. The learning rate is decayed to $0.99995\times$ of the previous iteration. Particularly, P-NeRF and NARF$_H$ based autoencoders are trained with an initial learning rate of 0.001 since the training will explode if a learning rate of 0.01 is used. The batch size is set to 16 for all experiments. We sample as many camera rays as can be fit in the GPU memory. The training converges at about 100,000 iterations. The training of our method NARF$_D$ takes 24 hours on 4 V100 GPUs. The code for creating our synthetic datasets will be released according to the acceptance of the paper.

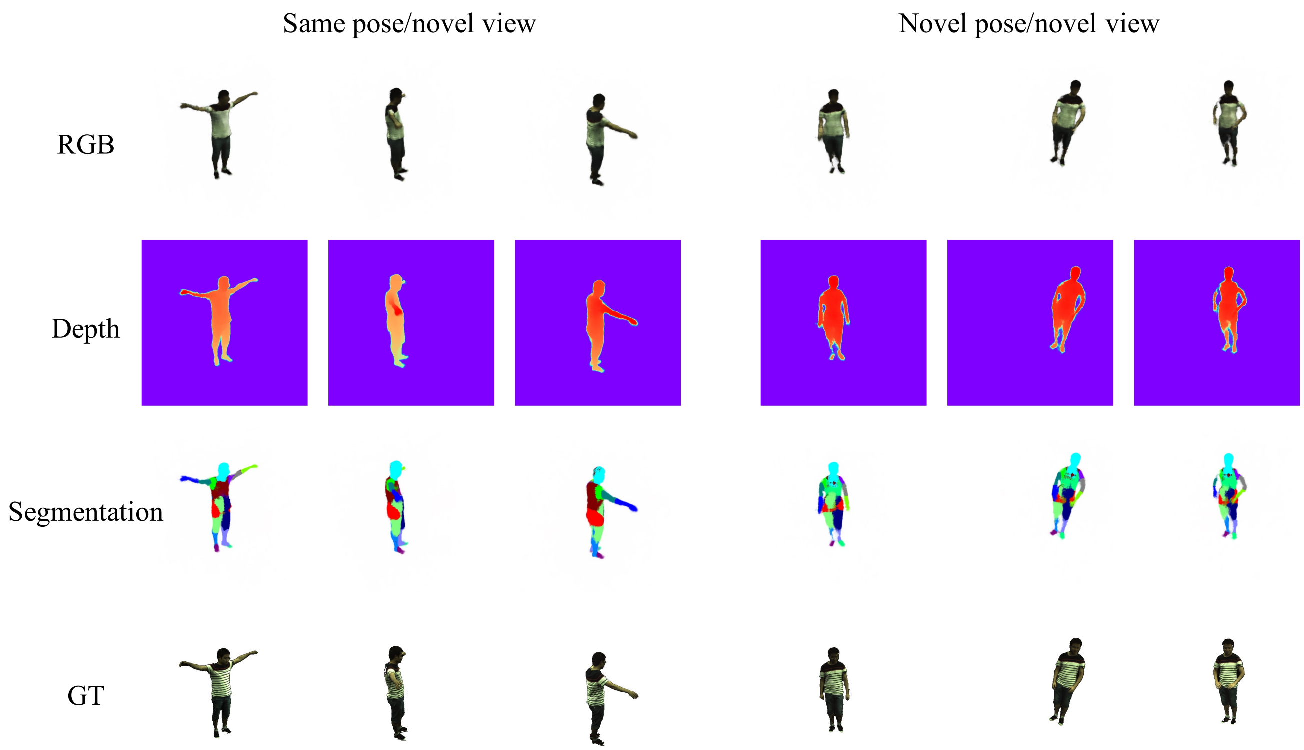

Figure 11. Results of NARF$_D$ on real human images

G. Experiment on Real Human Images

In this section, we test our approach on a real human dataset ZJU-MOCAP [46]. ZJU-MOCAP is a multi-view person video dataset. For each frame, SMPL parameters [35] are given. We use the first 1969 frames (90%, 2185 frames in total) of the Taichi class video for training and the remaining 216 frames (10%) for testing (novel pose). The resolution of the image is 512×512.

The qualitative results of NARF$_D$ on this dataset are shown in Figure 11. The left part of Figure 11 shows the pose used in the training, but rendered from novel viewpoints, and the right part of Figure 11 shows the novel pose/novel view testing results. The quality of the rendered images is not as good as testing on our synthetic datasets. This might be caused by the assumption that the parts are rigid objects, which may not be perfectly satisfied for real images. For example, loose clothes may move when a person makes a movement. This issue can be considered in future work, for example, by learning latent variables to account for both pose-dependent and pose-independent deformations similar to Neural Body [46].

Although the quality of the rendered images for a real person from our method still has room for improvement, we believe that the proposed explicitly controllable representation of viewpoint, pose, bone parameters, and appearance for the articulated object is an important contribution.