Neural Texture Puppeteer: A Framework for Neural Geometry and Texture Rendering of Articulated Shapes, Enabling Re-Identification at Interactive Speed

Urs Waldmann Ole Johannsen Bastian Goldluecke

University of Konstanz, Germany
{firstname.lastname}@uni-konstanz.de

Abstract

In this paper, we present a neural rendering pipeline for textured articulated shapes that we call Neural Texture Puppeteer. Our method separates geometry and texture encoding. The geometry pipeline learns to capture spatial relationships on the surface of the articulated shape from ground truth data that provides this geometric information. A texture auto-encoder makes use of this information to encode textured images into a global latent code. This global texture embedding can be efficiently trained separately from the geometry, and used in a downstream task to identify individuals. The neural texture rendering and the identification of individuals run at interactive speeds. To the best of our knowledge, we are the first to offer a promising alternative to CNN- or transformer-based approaches for re-identification of articulated individuals based on neural rendering. Realistic looking novel view and pose synthesis for different synthetic cow textures further demonstrate the quality of our method. Restricted by the availability of ground truth data for the articulated shape's geometry, the quality for real-world data synthesis is reduced. We further demonstrate the flexibility of our model for real-world data by applying a synthetic to real-world texture domain shift where we reconstruct the texture from a real-world 2D RGB image. Thus, our method can be applied to endangered species where data is limited. Our novel synthetic texture dataset NePuMoo is publicly available to inspire further development in the field of neural rendering-based re-identification.

1. Introduction

Recent developments in neural rendering brought a big boost to many vision applications [39, 40, 49]. One of them is novel view and novel pose synthesis. With the success of NeRF [26] that can render rigid objects in 3D from multiple input images, many other methods for novel view synthesis of static content were developed [20, 28]. Next, approaches that additionally handle articulated shapes leveraging the SMPL mesh model [23] were developed [30, 31]. For example, [47] generates an UV texture map that is combined with a rendering tensor generated with OpenDR [24] from the SMPL [23] model. A drawback of all methods that leverage the SMPL model [23] is that it limits the methods to a pre-defined class of shapes. Articulated shapes can also be handled with methods that leverage implicit neural representations [38] and do not rely on the SMPL model.

However, NeRF-based approaches use volumetric rendering which is time and memory intensive. In contrast, there are methods for static content [37] and articulated shapes [8] that render in 2D. Their advantage compared to volumetric rendering of NeRF-based approaches is that they require only a single neural network evaluation per ray. This results in significantly faster inference speed [8, 37].

Like neural rendering, tracking is a vast field of research and its applications range from self-driving cars [22] to the study of collective behaviour [15]. Therefore reliable and accurate tracking of objects like humans [3, 5, 33, 34] and animals [18, 32, 42–45] is required. An essential extension for many tracking applications is the re-identification of individuals when leaving and re-entering the scene. Most frameworks use CNNs or vision transformers to extract image features for object re-identifiaction [11, 18, 21, 52].

Surprisingly, only one study that we are aware of employs neural rendering to identify objects in a tracking scenario, namely [51], which tracks single rigid objects in motion in a self-supervised manner. It however has limitations in practical applications because it handles only rigid objects and has a slow inference speed as it is NeRF-based [26] and uses time- and memory-intensive volumetric rendering.

In particular for the study of collective behaviour, but also for other research in biology directed towards animals, it is crucial to have high-performance tracking [2] and re-identification of individuals [7]. The aim of our work is therefore to leverage the recent advances in neural render-

ing and develop a pipeline that can handle more than one textured articulated shape in a single model, thus enabling re-identification in tracking scenarios.

Contributions. We present a flexible neural rendering pipeline for textured articulated shapes that we call Neural Texture Puppeteer (NeTePu). Our method separates geometry and texture encoding. The geometry pipeline learns to capture spatial relationships on the surface of the articulated shape from ground truth data that provides this geometric information. For this purpose we extend the idea in [46] to articulated shapes, cf. Sec. 3.1. We show that our method encodes a distinct global texture embedding (cf. Fig. 5a), which is computed from this geometric information and 2D color information. This global texture embedding can be used in a downstream task to identify individuals.

Our neural rendering-based re-identification of individuals runs at interactive speeds (cf. Sec. 5.4) and thus offers a promising alternative to CNN- or transformer-based approaches in tasks such as the re-identification of individuals in tracking scenarios. To the best of our knowledge, we are the first to provide a framework for re-identification of articulated individuals based on neural rendering. We also demonstrate the quality of our method with realistic looking novel view and pose synthesis for different synthetic cow textures, cf. Fig. 3. We further demonstrate the flexibility of our model by applying a synthetic to real-world texture domain shift where we reconstruct the texture from a real-world 2D RGB image with a model trained on synthetic data only. This makes our model useful for real-world applications with endangered animal species.

Our novel synthetic texture dataset NePuMoo (cf. Sec. 4) together with the code to reproduce the results of this paper are publicly available at `https://github.com/urs-waldmann/NeTePu` to inspire further development in the field of neural rendering-based re-identification.

2. Related Work

We give an overview on re-identification, the texture prediction for articulated objects and datasets for these tasks. For an overview on neural rendering we refer to [39, 40, 49].

Texture Prediction of Articulated Objects. There exist a wide range of methods to predict the texture of articulated human shapes. Good results can be achieved with NeRF-based [26] approaches [30, 38], approaches based on implicit neural representations [31, 36] or approximate differentiable rendering [47]. In contrast to our method, all these approaches make some strong simplifications. While [30, 31, 47] leverage the SMPL [23] model, [30, 38] learn only one model per texture. Furthermore, [30, 36, 38] render in 3D, while our method shifts rendering to 2D by employing techniques from [8], which makes our method much faster.

For animals, we are aware of four works [14, 19, 53, 54] which learn UV maps which they project on a mesh model. In contrast to our neural rendering approach, they all use morphable models on which they map the predicted texture.

Only recently Wu *et al*. published [48]. This framework reconstructs textured articulated objects from single images. In contrast to this work, we extract NNOPCS maps (similar to NOCS maps [46], cf. Sec. 3.1) instead of a prior mesh for the object category. Further, instead of a collection of single-view images for an object category, we leverage a multi-view setup to assure that the individual's texture has been observed completely during training. This guarantees a meaningful global texture latent code for each individual from any viewpoint for re-identification in tracking tasks.

Re-identification of Individuals. Tracking is a vast field under constant development. For a summary, we refer to [4, 6, 50]. Re-identification of individuals is an important task in scenarios where individuals exit and re-enter the scene. Popular pipelines for object re-identification train CNN backbones or vision transformers to extract image features [11, 18, 21, 52], and often [11, 21, 52] need positive and negative pairs of the object in their training phase. Notably, [11, 18, 21, 52] base re-indentification on selected image features and do not make sure to encode the whole texture of the individual. In contrast, our encoded global latent code is used to reconstruct the complete texture from arbitrary viewpoints within our neural texture rendering pipeline.

From a single input image, [47] generates an UV texture map that is combined with a rendering tensor generated with OpenDR [24] using the SMPL [23] mesh model. In the end, they use a pre-trained person re-identification model to extract features of the rendered and the input image on which they calculate a re-identification loss. In contrast to our work, the authors use a re-identification network to generate a texture, while we use the texture for re-identification, which is a fundamental technical difference. Due to this, we can not perform an experimental comparison with [47].

The only other work that we aware of that uses neural rendering for tracking is [51]. While [51] reconstructs single rigid objects in motion from multi-view RGB videos in a self-supervised manner with a NeRF-based [26] neural rendering method, we can reconstruct articulated objects, which makes our method applicable to humans and animals. Furthermore our method shifts rendering from 3D to 2D, as it is based on the neural rendering framework [8]. This fundamental difference makes our method much faster compared to NeRF-based [26] approaches like [51].

Datasets. We aim to reconstruct textures of articulated shapes with a keypoint-based neural rendering pipeline that leverages our NNOPCS maps. For this task we need datasets that contain articulated shapes like humans or animals together with keypoints, segmentation masks, RGB images and NNOPCS maps. For humans the most popular real-world dataset is H36M [13]. For animals we are

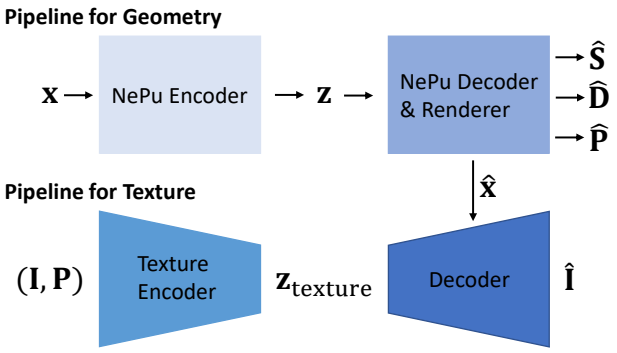


Figure 1. **Complete Pipeline.** Our key idea is to disentangle geometry and texture information of articulated shapes. We modify the original NePu [8] pipeline to predict a NNOPCS map $\hat{\mathbf{P}}$ (top). We also predict the silhouette $\hat{\mathbf{S}}$ and depth map $\hat{\mathbf{D}}$ like the original NePu pipeline. The heart of our framework is an encoder-decoder network for the texture of an articulated shape (bottom) that reconstructs an image $\mathbf{I}$. Details and the model's loss are in Sec. 3.

aware of four datasets containing more than one individual [1, 10, 25, 27]. All these datasets lack ground truth NNOPCS maps for the articulated shapes they contain. That is why we create a novel synthetic dataset of textured articulated shapes that provides NNOPCS map ground truth. For a proof of concept, we choose cows as an example species because their textures can be clearly distinguished.

3. Architecture

The heart of our framework is an encoder-decoder network that reconstructs the texture of an articulated shape, cf. Fig. 1. As a basis, we make use of Neural Puppeteer (NePu) [8] for neural rendering.

NePu is a neural rendering pipeline designed specifically for articulated shapes, which takes as input a set of K 3D keypoints $\mathbf{x} \in \mathbb{R}^{K\times 3}$ and a camera position, and maps them to a RGB image of the shape (similar to Fig. 1, top: Pipeline for Geometry). Notably, NePu renders in the 2D domain, which makes it much faster than volumetric rendering: via a transformer network, local features are computed for each keypoint (NePu Decoder in Fig. 1, top), which are then projected onto the image plane by the camera. Another attention-based network, the actual renderer, then generates a complete 2D image from the projected features at the keypoint locations (NePu Renderer in Fig. 1, top).

Our key idea in this work is to disentangle the pipelines for geometry and texture information of the shape that we call **Neural Texture Puppeteer (NeTePu)**. Thus NeTePu, unlike NePu and [30, 38], can handle more than one texture in a single model. The original NePu pipeline is therefore modified to produce what we call a NNOPCS map (cf. Sec. 3.1) instead of a final image, cf. Fig. 1 (top). This NNOPCS map essentially encodes within the image plane the complete geometric information about the shape relevant for rendering this particular view. Local features $\mathbf{f}$ (we refer to [8] and Sec. 3.1 for details) and global latent code $\mathbf{z}$ (cf. Fig. 1, top) for the geometry are augmented with separate local features $\mathbf{f}_{\text{texture}}$ and a latent code $\mathbf{z}_{texture}$ for the texture map. An encoder network generates $\mathbf{z}_{texture}$ for a texture from a view of the shape, while the decoder is a second NePu-based renderer to generate the final image (cf. Fig. 1, bottom). Putting together the encoder and decoder for $\mathbf{z}_{texture}$, we essentially obtain a texture autoencoder which, given a fixed geometry, can be efficiently trained separately from the geometry network.

In the following subsections, we will give the details for these different modules.

3.1. Pipeline for Geometry

In our framework, the geometry and pose of an articulated shape is defined by the K different keypoint locations and a single global latent code $\mathbf{z} \in \mathbb{R}^{d_z}$ for the shape, which is decoded into a d_f-dimensional local feature vector $\mathbf{f}$ for each keypoint as input to the renderer. By varying the keypoint locations and keeping the features fixed, one can generate renderings of arbitrary poses of the same geometry [8]. We now use the rendering pipeline in [8], but do not interpret the output as a rendered color image. Instead, at each point in the image plane, the 3D output defines a 3D coordinate of a point on the shape in a normalized coordinate space (NOCS [46]), and additionally in a normalized neutral pose, cf. Fig. 2a, which is why we denote it **N**ormalized **N**eutral **O**bject **P**ose and **C**oordinate **S**pace (NNOPCS). Thus, the same point on the shape always has the same encoding, no matter the pose of the object and the camera view. The NNOPCS map can thus be viewed as a generalization of a NOCS map [46].

NNOPCS Maps for Articulated Shapes. Similar to [8], a coordinate on the object is defined via a 3D vector for each pixel of the target view with width W and height H, thus the NNOPCS map is a function of the form

$$\mathcal{P}_{\mathbf{E},\mathbf{K}} : \mathbb{R}^{K\times 3} \times \mathbb{R}^{K\times d_f} \times \mathbb{R}^{d_z} \rightarrow [0,1]^{H\times W\times 3}. \quad (1)$$

Here, $\mathbf{E}, \mathbf{K}$ are the given camera extrinsics and intrinsics, respectively. The architecture remains the same, so we refer to [8] for details on how the function is implemented as an attention-based neural network. Similar to [46], the interpretation of the 3D output of the function in Eq. (1) is a dense pixel-NNOPCS correspondence. Ground truth data to learn the NNOPCS maps for articulated shapes end-to-end is rendered via Blender (www.blender.org), see section 4.

3.2. Pipeline for Texture

Texture Encoder. The texture of a shape is described by a d_z-dimensional global texture latent code $\mathbf{z}_{texture}$ (same

dimension as $\mathbf{z}$), which augments the global geometry latent code $\mathbf{z}$, cf. Fig. 1. An encoder network enc computes $\mathbf{z}_{texture}$ from a masked input image $\mathbf{I}$ together with the NNOPCS map $\mathbf{P}$ (cf. Sec. 3.1) for a certain camera position,

$$enc : \mathbb{R}^{H\times W\times 3} \times \mathbb{R}^{H\times W\times 3} \to \mathbb{R}^{d_z}, \quad (\mathbf{I}, \mathbf{P}) \mapsto \mathbf{z}_{texture} \tag{2}$$

The texture encoder is implemented as a convolutional neural network with 23 convolution and two dense layers. Between the layers we use batch normalization [12] and ReLu activations. Similar to a variational autoencoder [16], the texture encoder outputs a mean $\mu \in \mathbb{R}^{d_z}$ and standard deviation $\sigma^2 \in \mathbb{R}^{d_z}$ from which we sample the global texture latent code $\mathbf{z}_{texture}$.

During training, we use the available ground truth NNOPCS maps (cf. Sec. 3.1) and silhouette masks. During inference, the silhouette mask and the NNOPCS map is estimated with the network trained above, cf. Fig. 1. Both the images as well as the NNOPCS maps are set to zero outside the shape's silhouette.

Neural Texture Rendering. Similar to how we render the NNOPCS maps (cf. Sec. 3.1), we first decode the global texture embedding $\mathbf{z}_{texture}$ from Eq. (2) into local texture features with a decoder network

$$dec : \mathbb{R}^{d_z} \to \mathbb{R}^{K\times d_f},\ \mathbf{z}_{texture} \mapsto \mathbf{f}_{texture}, \tag{3}$$

where $\mathbf{f}_{texture}$ gives the d_f-dimensional local texture features for each keypoint. Note that the dimension of $\mathbf{f}_{texture}$ is the same as the local geometry features $\mathbf{f}$, so the final renderer for a 2D image seen from camera view c conditioned on $\mathbf{z}_{texture}$ and keypoints $\mathbf{x}$ is again (compare with Eq. (1)) a function of the form

$$\mathcal{C}_{\mathbf{E},\mathbf{K}} : \mathbb{R}^{K\times 3} \times \mathbb{R}^{K\times d_f} \times \mathbb{R}^{d_z} \to [0,1]^{H\times W\times 3}. \tag{4}$$

4. Datasets and Training

NePuMoo Dataset. For this proof of concept, we choose cows as an example shape because their textures can be clearly distinguished. We extend [9] with its Holstein cow texture by adding eleven additional cow textures. For each texture we provide the same 910 poses, each captured from the same 24 perspectives, placed in three evenly sampled rings at different heights around the model. This results in 21840 views per texture and a total of 262080 views. Each view consists of the ground truth 3D and 2D keypoints, the rendered RGB image, a silhouette of the cow, a depth map and a NNOPCS map, cf. Sec. 3.1. Each view has a resolution of 1024×768 px. All images were rendered using Blender (www.blender.org) and the Cycles rendering engine. We provide the same 25 keypoints distributed among the joints of the cow as [9].

Our synthetic texture dataset also contains instances of occlusions. We generated 50 instances captured from two of the 24 camera views. In these samples the Holstein (texture 0) partially occludes the Limousine (texture 11) cow. For these cases we provide the occluded and complete silhouettes.

H36M Dataset. This human real-world dataset contains keypoint annotations, segmentations masks and RGB images of eleven actors in 17 scenarios, cf. [13]. We use the first 910 frames of the "Posing" scenario from 7 subjects (S1, S5, S6, S7, S8, S9, S11) for this study, similar to [30], and all four camera views for training.

Training the Pipeline for Geometry. To train the NNOPCS maps of articulated shapes, cf. Sec. 3.1, we use the same regimen as in [8]. However, instead of color images we train end-to-end with the ground truth NNOPCS map observations

$$\mathbf{P}_{m,c},\ m \in \{1,\dots,M\},\ c \in \{1,\dots,C\} \tag{5}$$

over M distinct poses captured by C different cameras. All model parameters are trained jointly. In the composite loss from [8], instead of the color loss $\mathcal{L}_{\text{col}}$ we define

$$\mathcal{L}_{\text{NNOPCS}} = \sum_{m=1}^{M}\sum_{c=1}^{C} \|\mathcal{P}_{\mathbf{E}_c,\mathbf{K}_c} - \mathbf{P}_{m,c}\|_2^2, \tag{6}$$

which is the squared pixel-wise difference over all three channels. For the details of this equation and the composite loss, we refer to [8].

Training the Pipeline for Texture. While training our neural texture encoder, decoder and renderer (cf. Fig. 1, bottom), we keep the weights of the pipeline for geometry (cf. Fig. 1, top) fixed and use the ground truth NNOPCS as the fixed geometry. We thus learn only the texture relevant weights of the pipeline in this step.

Assuming the dataset has T textures with images $\mathbf{I}_{t,m,c}$ provided for each texture and the same poses and cameras as the NNOPCS maps, we define the color rendering loss

$$\mathcal{L}_{\text{col}} = \sum_{t=1}^{T}\sum_{m=1}^{M}\sum_{c=1}^{C} \|\mathcal{C}_{\mathbf{E}_c,\mathbf{K}_c} - \mathbf{I}_{m,c,t}\|_2^2 \tag{7}$$

as the squared pixel-wise difference over all color channels, where $\mathcal{C}_{\mathbf{E},\mathbf{K}}$ is the color rendering function from [8] defined in Eq. (4). To regularize the texture latent space, we employ the Kullback–Leibler divergence [17] $\mathcal{L}_{\text{KLD}}$ between the predicted multi-dimensional normal distribution and the standard normal distribution. We enforce this loss with the same sampling scheme and reparametrization trick as in the training of a variational autoencoder [16]. The total loss is thus

$$\mathcal{L} = \lambda_{\text{col}}\mathcal{L}_{\text{col}} + \lambda_{\text{KLD}}\mathcal{L}_{\text{KLD}}, \tag{8}$$

where the different positive numbers λ are hyperparameters to balance the influence of the two losses.

NNOPCS Maps MSE	Depth MAE [mm]		
Ours	NePu [8]	Ours	Δ
$1.6 \cdot 10^{-4}$	22.3	**17.8**	−4.5

Table 1. *Quantitative results for the NNOPCS and depth maps on the NePuMoo test set.* We report MSE for the reconstructed NNOPCS maps (cf. Sec. 3.1) and MAE [mm] for the reconstructed depth maps. Comparison of the reconstructed depth map between NePu [8] and our method. See Sec. 5.2 for a discussion of the results.

5. Experiments

In Sec. 5.1 we provide implementation details of our architecture from Sec. 3. We then evaluate the learned NNOPCS maps (cf. Sec. 3.1) in Sec. 5.2. In Sec. 5.3 we show novel pose synthesis on our novel NePuMoo and the H36M [13] dataset and provide texture reconstructions of a real-world example with a model trained on synthetic data only. In Sec. 5.4 you find our kernel density estimation (KDE) [29, 35] of the t-SNE [41] of the global texture embedding that we compare to the embedding of [11].

5.1. Implementation Details

To train NeTePu, we extend the training procedure and parameters from [8]. We do not render complete NNOPCS (cf. Sec. 3.1) and texture maps to compute $\mathcal{L}_{\text{NNOPCS}}$ and $\mathcal{L}_{\text{col}}$ during training, as this would be too costly. Instead, we choose 500 points in each iteration that we sample uniformly within the ground truth mask. For the cow shape we increase the number of nearest neighbours in the vector cross-attention (VCA) module from 12 to 20 as we observe better performance (cf. Tab. 1).

Pipeline for Geometry. While training the NNOPCS maps, we also learn to reconstruct masks and depth, just as in [8]. However, in contrast to [8], we choose to use the 2-norm (vs. MSE in [8]) as depth loss during training which leads to better results in terms of MAE, cf. Tab. 1.

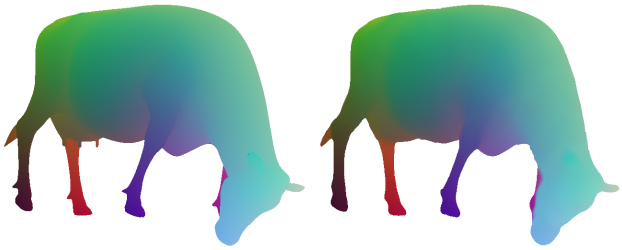
Pipeline for Texture. We train the neural texture rendering pipeline using an initial learning rate of $5e^{-4}$, which is decayed with a factor of 0.2 on our novel NePuMoo dataset and 0.9 on the H36M dataset [13] every 500 epochs. We choose final weights based on the minimum validation loss, i.e. epoch 535 for our NePuMoo and 2080 for the H36M dataset. We weight the training loss in Eq. (8) with $\lambda_{\text{col}} = 5$ and $\lambda_{\text{KLD}} = 1e^{-8}$.

5.2. NNOPCS Maps

Quantitative results for the NNOPCS map (cf. Sec. 3.1) estimation are shown in Tab. 1. Over all cow test samples, we achieve a mean squared error (MSE) of $1.6 \cdot 10^{-4}$. Since the coordinates of the NNOPCS are normalized to $[0, 1]$ this means that our learned NNOPCS maps have an error of 1% on average. This overall high quality is also reflected in our qualitative example in Fig. 2b, and is a necessary step in order to achieve accurate texture renderings.

In Tab. 1, we also compare depth map reconstructions of NeTePu to the original NePu [8] in terms of the mean absolute error (MAE) in mm over all cow test samples. We achieve a lower MAE by $4.5mm$, compared to the length and height of the cow of $2.2m$ and $1.65m$, respectively.

(a) **Neutral Pose.** The color at each point on the mesh depends on its position in x-, y-, and z-direction in the neutral pose. Thus, for other poses the color at a specific position on the mesh remains the same, uniquely describing this point.



(b) **Novel Pose Synthesis.** One example of ground truth (left) and reconstruction (right) from our NePuMoo test set. Our reconstruction (right) reflects the overall high quality of our quantitative results in Tab. 1.

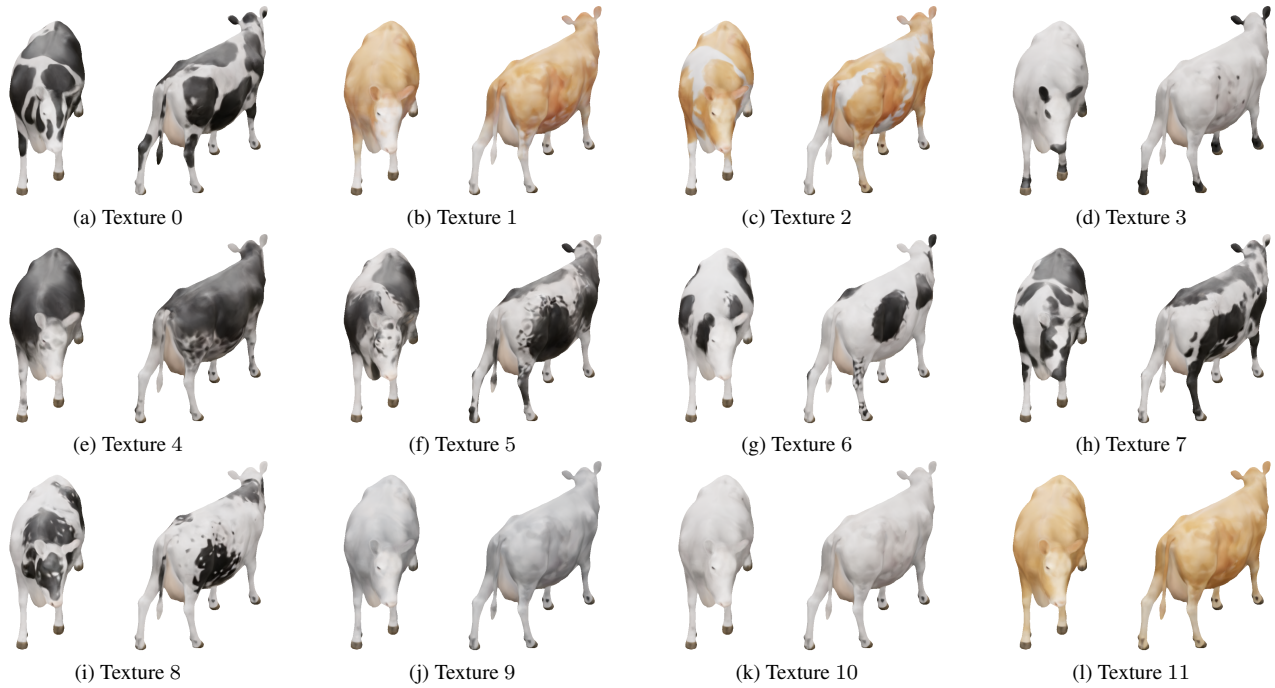
Figure 2. **NNOPCS Maps for Articulated Shapes.** We show the neutral pose of the NNOPCS (cf. Sec. 3.1) projected on two image planes in (a) and one example for a novel pose from our NePuMoo test set in (b).

Dataset / Metric	Color PSNR [dB]
H36M [13]	15.98
NePuMoo	19.35

Table 2. *Quantitative results for novel pose synthesis.* We report the PSNR [dB] for the reconstructed RGB images on the real-world H36M [13] and our NePuMoo test set, cf. Sec. 4. See Sec. 5.3 for a discussion of the results.

5.3. Neural Texture Rendering

Quantitative results for novel pose synthesis are shown in Tab. 2. We report the PSNR [dB] over all test samples for our reconstructed RGB images on the real-world H36M [13] and our NePuMoo test set.



(a) Texture 0 (b) Texture 1 (c) Texture 2 (d) Texture 3

(e) Texture 4 (f) Texture 5 (g) Texture 6 (h) Texture 7

(i) Texture 8 (j) Texture 9 (k) Texture 10 (l) Texture 11

Figure 3. **Novel Pose and Novel View Synthesis.** We reconstruct images for novel views and novel poses from our NePuMoo test set, which have not been seen during training. We show two examples for each distinct texture in our dataset. Our reconstructions reflect the overall good quality of our quantitative results in Tab. 2.

We achieve an overall PSNR of $19.35dB$ on our novel NePuMoo data. This is slightly better than the PSNR of $19.17dB$ reported in [8] although in contrast to [8], we render twelve textures instead of one with a single model.

On the H36M data [13] we achieve an overall PSNR of $15.98dB$. See Fig. 6 (right pair) for a qualitative example. Please note that the H36M data does not provide NNOPCS maps (cf. Sec. 3.1) and we thus train NeTePu without them. In this case NeTePu can not leverage the geometric shape information provided by the NNOPCS maps. This holds for any data that does not provide NNOPCS maps. For a deeper insight on the influence and the importance of the NNOPCS maps, we refer to our ablation studies.

We reconstruct images for novel views and novel poses from our NePuMoo test set, which have not been seen during training. In Fig. 3 we show two examples for each distinct texture in our dataset. The reconstructions look realistic and contain details like the black freckles of texture $3-6$ and 8 (cf. eg. Fig. 3f) or the black legs, ears and nose tip of texture 3 (cf. Fig. 3d). The dataset contains two challenging texture pairs, which are difficult to distinguish by eye. We note that even these challenging texture pairs 9, 10 and 1, 11 are reconstructed correctly (cf. Figs. 3j and 3k and Figs. 3b and 3l respectively). We also perform a texture domain shift from synthetic to real. For this, we use the model that we train on our novel NePuMoo texture dataset and infer a zero-shot synthetic to real-world example. Our reconstructions are shown in Fig. 4. We reconstruct the texture of the real-world image for three different poses (cf. Fig. 4b). Even though only trained with synthetic data, this shows that NeTePu can be used for real-world applications that do not provide NNOPCS maps, cf. Sec. 3.1. This makes our model useful for real-world applications with endangered animal species where available real-world data is limited.

	Color PSNR [dB]
w/o NNOPCS Maps	13.63
w/ NNOPCS Maps	**19.35**

Table 3. *Ablation study on the influence of the NNOPCS maps.* We report the PSNR [dB] for the reconstructed RGB images on our NePuMoo test set, cf. Sec. 4. We report results for an architecture that leverages the NNOPCS maps (cf. Sec. 3.1) and one that does not. Best result is bold.

Ablation Studies. To evaluate the influence of the NNOPCS maps (cf. Sec. 3.1) we run an experiment on our novel NePuMoo dataset with the same parameters as specified in Sec. 5.1. The only difference is that we do not use NNOPCS maps in our pipeline, cf. Fig. 1. We report the color PSNR over all test samples in Tab. 3. If our model can leverage geometric information of the shape that is provided by the NNOPCS maps, NeTePu achieves a better color PSNR on average by $5.72dB$. This quantitative dif-

(a) Masked Real-World Image

(b) Our Reconstructions

Figure 4. **Synthetic to Real-World Texture Domain Shift.** We show a zero-shot synthetic to real-world example. NeTePu reconstructs the texture from a real-world RGB image. We show the reconstructed texture in three different poses.

ference is visible in the qualitative results, cf. Fig. 6 (left pair). This also explains the quality of our results on the H36M dataset [13], cf. Tab. 2 and Fig. 6 (right pair).

We perform another ablation study on the influence of the local texture features $\mathbf{f}_{\text{texture}} \in \mathbb{R}^{d_f}$. We run an experiment on our novel NePuMoo data where we do not augment the local features for geometry $\mathbf{f}$ with local features for texture $\mathbf{f}_{\text{texture}}$. We see that augmenting the local features for geometry $\mathbf{f}$ with $\mathbf{f}_{\text{texture}}$ achieves a better color PSNR on average by 4.61 dB (19.35 dB in Tab. 2 vs. 14.74 dB).

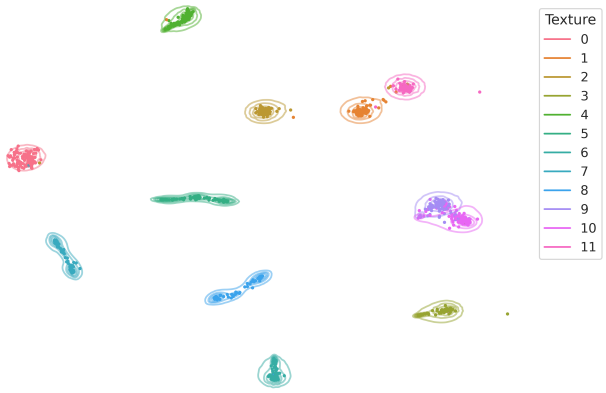
5.4. Re-identification

Our proposed neural rendering pipeline learns a global texture embedding which can be used to identify individuals. This is a valuable task for applications that need re-identification of individuals like tracking scenarios where the tracked object leaves and re-enters the scene. For this task, we use Eq. (2) to encode a global latent vector $\mathbf{z}_{texture}^{m,c,t}$ that describes the texture of individual t observed from camera view c in pose m. In this way, we can generate a global texture embedding $\mathbf{Z}_{texture}$ (capital Z) for all individuals under consideration from our learned NNOPCS maps (cf. Sec. 3.1) $\hat{\mathbf{P}}_{m,c}$ and masked 2D color observations $\tilde{\mathbf{I}}_{m,c,t}$,

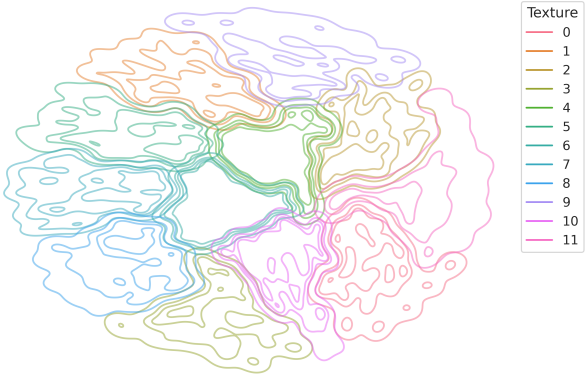
$$\mathbf{Z}_{texture} = \{\mathbf{z}_{texture}^{m,c,t}\}, \\ m \in \{1,\dots,M\},\ c \in \{1,\dots,C\},\ t \in \{1,\dots,T\}. \tag{9}$$

Baseline. For a comparison, we train TransReID [11] on our novel NePuMoo data, cf. Sec. 4. [11] is a transformer-based framework for object re-identification that is optimized with ID [52] and triplet loss [21]. During training [11] needs positive and negative pairs of the object. We resize the input to 128×256 px and use a batch size of 40. For the other parameters we use their standard. After training, we generate a global feature embedding $\mathbf{Z}$ with their global features for a comparison.

Results. In Fig. 5a we show NeTePu's kernel density estimation (KDE) [29, 35] of the t-SNE [41] of the global texture embedding from Eq. (9) of twelve cows from our novel synthetic NePuMoo dataset (cf. Sec. 4). We encode the global texture codes of all 910 poses, twelve textures and 24 cameras. While the contour lines show the distribution of global texture codes for the camera views from our



(a) NeTePu: In addition to the distribution of latent codes for known camera views from the training set (contour lines), the dots show novel views and fall into the clusters of the known ones.



(b) TransReID [11]: While the embedding is more compact, the individual clusters overlap more often.

Figure 5. **KDE of the Global Texture Embedding.** We show the KDE [29,35] of the t-SNE of the global texture codes. The contour lines show the distribution of latent codes for camera views from our novel NePuMoo dataset. The individual cows cluster nicely. See Sec. 5.4 for a discussion of the results.

Figure 6. **Novel Pose Synthesis Without NNOPCS Maps.** Color reconstruction and ground truth for a novel pose, where the pipeline does not use NNOPCS maps (cf. Sec. 3.1). *Left Pair*: Holstein cow (texture 0) from our NePuMoo test set, cf. Sec. 4. *Right Pair*: Subject 8 from the H36M dataset [13]. See Secs. 5.3 and 6 for a discussion of the results.

dataset, the dots show the global codes for novel camera views. In Fig. 5b we show the KDE of the global feature embedding $\mathbf{Z}$ generated with [11] for a comparison.

We see that the textures cluster in such a way that they can be distinguished by both frameworks. Both frameworks can distinguish the challenging textures 1 and 11 (cf. Figs. 3b and 3l) and 9 and 10 (cf. Figs. 3j and 3k). While the embedding of [11] is more compact, we observe only one slight overlap of texture clusters for NeTePu. No cluster overlap in the embedding helps in readily distinguishing all textures in all poses from all camera views. Furthermore, note that NeTePu's global texture latent codes of novel camera views (dots in Fig. 5a) fall into the corresponding clusters of the known camera views (contour lines in Fig. 5a).

Runtime. We evaluate the runtime on our novel NePuMoo dataset with a workstation that has a nVidia Titan RTX, 64 GB DDR4 RAM, an Intel Xeon E5-2620 at 2.10GHz and a 2TB Samsung SSD 850. We encode the global texture code from an input of size $768 \times 1024 \times 6$ (masked 2D color observation and learned NNOPCS map, cf. Sec. 3.1). This calculation includes to render our learned NNOPCS maps with a resolution of $768 \times 1024 \times 3$. Since our NNOPCS maps are only defined within the object, the calculation also includes to render the learned mask with a resolution of 768×1024. We use a batch size of 1 for all twelve textures of 100 samples from our dataset seen from 24 camera views. In total, we thus encode the global texture code for 28800 frames. To calculate the runtime, we take the average over these frames. We achieve a runtime of 0.5 fps.

All together, this is why our proposed neural rendering pipeline offers an alternative approach to CNN- or transformer-based frameworks like [11] for the task of re-identification in interactive tracking applications.

6. Limitations and Future Work

At the moment, we learn the NNOPCS maps in a supervised manner, cf. Sec. 3.1. This limits our method to data that provide NNOPCS maps. If we train on data that does not provide these NNOCPS maps, the quality of the outcome diminishs because the model lacks information on the geometry's shape, cf. Fig. 6. In order to make our method more applicable, especially to real-world applications, we aim to learn the NNOPCS maps in the future in an unsupervised manner. Also, at this point our dataset contains a limited variety of textures. In the future we aim at extending our dataset to allow for better generalizability when it comes to new - unseen - textures.

7. Conclusions

In this paper, we present a neural rendering pipeline for textured articulated shapes. We show that NeTePu encodes a distinct global texture embedding (cf. Fig. 5a), which is computed from a learned NNOPCS map (cf. Sec. 3.1) and 2D color information. This global texture embedding can be used in a downstream task to identify individuals. Our neural rendering-based re-identification process runs at interactive speed (cf. Sec. 5.4), and thus offers an alternative to CNN- or transformer-based approaches like [11] in tracking applications. To the best of our knowledge, we are the first to provide a framework for re-identification of articulated individuals based on neural rendering. We also demonstrate NeTePu's quality with realistic looking novel view and pose synthesis for different synthetic cow textures, cf. Fig. 3. Restricted by the availability of ground truth NNOPCS maps, the quality for real-world data synthesis is reduced, cf. Fig. 6 (right pair). We further demonstrate the flexibility of our model by applying a synthetic to real-world texture domain shift where we reconstruct the texture from a real-world 2D RGB image with a model trained on synthetic data only. This makes our model useful for real-world applications with endangered animal species where available real-world data is limited and synthetic data can be generated using Blender (`www.blender.org`).

We hope that this work inspires other researchers to develop methods for neural rendering-based re-identification which work for humans and animals.

Acknowledgements. We acknowledge funding by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) under Germany's Excellence Strategy – EXC 2117 – 422037984, and the Federal Ministry of Education and Research (BMBF) within the research program – KI4KMU – 01IS23046B.

References

[1] Praneet C. Bala, Benjamin R. Eisenreich, Seng Bum Michael Yoo, Benjamin Y. Hayden, Hyun Soo Park, and Jan Zimmermann. Automated markerless pose estimation in freely moving macaques with openmonkeystudio. *Nat. Commun.*, 11:4560, 2020. 3

[2] Alex Bewley, Zongyuan Ge, Lionel Ott, Fabio Ramos, and Ben Upcroft. Simple online and realtime tracking. In *ICIP*, pages 3464–3468, 2016. 1

[3] Long Chen, Haizhou Ai, Rui Chen, Zijie Zhuang, and Shuang Liu. Cross-view tracking for multi-human 3d pose estimation at over 100 fps. In *CVPR*, pages 3279–3288, 2020. 1

[4] Gioele Ciaparrone, Francisco Luque Sánchez, Siham Tabik, Luigi Troiano, Roberto Tagliaferri, and Francisco Herrera. Deep learning in video multi-object tracking: A survey. *Neurocomputing*, 381:61–88, 2020. 2

[5] Ying Cui, Dongyan Guo, Yanyan Shao, Zhenhua Wang, Chunhua Shen, Liyan Zhang, and Shengyong Chen. Joint classification and regression for visual tracking with fully convolutional siamese networks. *IJCV*, pages 1–17, 2022. 1

[6] Patrick Dendorfer, Aljosa Osep, Anton Milan, Konrad Schindler, Daniel Cremers, Ian Reid, Stefan Roth, and Laura Leal-Taixé. Motchallenge: A benchmark for single-camera multiple target tracking. *IJCV*, 129(4):845–881, 2021. 2

[7] André C Ferreira, Liliana R Silva, Francesco Renna, Hanja B Brandl, Julien P Renoult, Damien R Farine, Rita Covas, and Claire Doutrelant. Deep learning-based methods for individual recognition in small birds. *Methods in Ecology and Evolution*, 11(9):1072–1085, 2020. 1

[8] Simon Giebenhain, Urs Waldmann, Ole Johannsen, and Bastian Goldluecke. Neural puppeteer: Keypoint-based neural rendering of dynamic shapes. In *Proceedings of the Asian Conference on Computer Vision (ACCV)*, pages 2830–2847, December 2022. 1, 2, 3, 4, 5, 6

[9] Simon Giebenhain, Urs Waldmann, Ole Johannsen, and Bastian Goldlücke. Neural puppeteer: Keypoint-based neural rendering of dynamic shapes (datset), October 2022. 4

[10] Yaning Han, Ke Chen, Yunke Wang, Wenhao Liu, Xiaojing Wang, Jiahui Liao, Yiting Huang, Chuanliang Han, Kang Huang, Jiajia Zhang, et al. Social behavior atlas: A computational framework for tracking and mapping 3d close interactions of free-moving animals. *bioRxiv*, pages 2023–03, 2023. 3

[11] Shuting He, Hao Luo, Pichao Wang, Fan Wang, Hao Li, and Wei Jiang. Transreid: Transformer-based object re-identification. In *ICCV*, pages 15013–15022, 2021. 1, 2, 5, 7, 8

[12] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In Francis Bach and David Blei, editors, *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pages 448–456, Lille, France, 07–09 Jul 2015. PMLR. 4

[13] Catalin Ionescu, Dragos Papava, Vlad Olaru, and Cristian Sminchisescu. Human3.6m: Large scale datasets and predictive methods for 3d human sensing in natural environments. *IEEE TPAMI*, 36(7):1325–1339, 2014. 2, 4, 5, 6, 7, 8

[14] Angjoo Kanazawa, Shubham Tulsiani, Alexei A. Efros, and Jitendra Malik. Learning category-specific mesh reconstruction from image collections. In *ECCV*, September 2018. 2

[15] Roland Kays, Margaret C. Crofoot, Walter Jetz, and Martin Wikelski. Terrestrial animal tracking as an eye on life and planet. *Science*, 348(6240):aaa2478, 2015. 1

[16] Diederik P. Kingma and Max Welling. Auto-Encoding Variational Bayes. In *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*, 2014. 4

[17] Solomon Kullback and Richard A Leibler. On information and sufficiency. *The annals of mathematical statistics*, 22(1):79–86, 1951. 4

[18] Jessy Lauer, Mu Zhou, Shaokai Ye, William Menegas, Steffen Schneider, Tanmay Nath, Mohammed Mostafizur Rahman, Valentina Di Santo, Daniel Soberanes, Guoping Feng, Venkatesh N. Murthy, George Lauder, Catherine Dulac, Mackenzie W. Mathis, and Alexander Mathis. Multi-animal pose estimation, identification and tracking with deeplabcut. *Nat. Methods*, 19:496–504, 2022. 1, 2

[19] Xueting Li, Sifei Liu, Kihwan Kim, Shalini De Mello, Varun Jampani, Ming-Hsuan Yang, and Jan Kautz. Self-supervised single-view 3d reconstruction via semantic consistency. In Andrea Vedaldi, Horst Bischof, Thomas Brox, and Jan-Michael Frahm, editors, *ECCV*, pages 677–693, Cham, 2020. Springer International Publishing. 2

[20] Chen-Hsuan Lin, Chaoyang Wang, and Simon Lucey. Sdf-srn: Learning signed distance 3d object reconstruction from static images. *NeurIPS*, 33:11453–11464, 2020. 1

[21] Hao Liu, Jiashi Feng, Meibin Qi, Jianguo Jiang, and Shuicheng Yan. End-to-end comparative attention networks for person re-identification. *IEEE Transactions on Image Processing*, 26(7):3492–3506, 2017. 1, 2, 7

[22] Ze Liu, Yingfeng Cai, Hai Wang, Long Chen, Hongbo Gao, Yunyi Jia, and Yicheng Li. Robust target recognition and tracking of self-driving cars with radar and camera information fusion under severe weather conditions. *IEEE Transactions on Intelligent Transportation Systems*, 23(7):6640–6653, 2021. 1

[23] Matthew Loper, Naureen Mahmood, Javier Romero, Gerard Pons-Moll, and Michael J. Black. Smpl: A skinned multi-person linear model. *ACM Trans. Graph.*, 34(6), oct 2015. 1, 2

[24] Matthew M Loper and Michael J Black. Opendr: An approximate differentiable renderer. In *ECCV*, pages 154–169. Springer, 2014. 1, 2

[25] Jesse D Marshall, Ugne Klibaite, Amanda Gellis, Diego E Aldarondo, Bence P Ölveczky, and Timothy W Dunn. The pair-r24m dataset for multi-animal 3d pose estimation. *bioRxiv*, pages 2021–11, 2021. 3

[26] 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, 2

[27] Hemal Naik, Alex Hoi Hang Chan, Junran Yang, Mathilde Delacoux, Iain D. Couzin, Fumihiro Kano, and Máté Nagy. 3d-pop - an automated annotation approach to facilitate markerless 2d-3d tracking of freely moving birds with marker-based motion capture. In *CVPR*, pages 21274–21284, June 2023. 3

[28] Michael Niemeyer, Lars Mescheder, Michael Oechsle, and Andreas Geiger. Differentiable volumetric rendering: Learning implicit 3d representations without 3d supervision. In *CVPR*, pages 3504–3515, 2020. 1

[29] Emanuel Parzen. On Estimation of a Probability Density Function and Mode. *The Annals of Mathematical Statistics*, 33(3):1065 – 1076, 1962. 5, 7

[30] 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 *ICCV*, pages 14314–14323, October 2021. 1, 2, 3, 4

[31] 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. 1, 2

[32] Talmo D. Pereira, Nathaniel Tabris, Arie Matsliah, David M. Turner, Junyu Li, Shruthi Ravindranath, Eleni S. Papadoyannis, Edna Normand, David S. Deutsch, Z. Yan Wang, Grace C. McKenzie-Smith, Catalin C. Mitelut, Marielisa Diez Castro, John D'Uva, Mikhail Kislin, Dan H. Sanes, Sarah D. Kocher, Samuel S.-H. Wang, Annegret L. Falkner, Joshua W. Shaevitz, and Mala Murthy. Sleap: A deep learning system for multi-animal pose tracking. *Nat. Methods*, 19:486–495, 2022. 1

[33] Umer Rafi, Andreas Doering, Bastian Leibe, and Juergen Gall. Self-supervised keypoint correspondences for multi-person pose estimation and tracking in videos. In *ECCV*, pages 36–52. Springer International Publishing, 2020. 1

[34] Jathushan Rajasegaran, Georgios Pavlakos, Angjoo Kanazawa, and Jitendra Malik. Tracking people by predicting 3d appearance, location and pose. In *CVPR*, pages 2740–2749, 2022. 1

[35] Murray Rosenblatt. Remarks on Some Nonparametric Estimates of a Density Function. *The Annals of Mathematical Statistics*, 27(3):832 – 837, 1956. 5, 7

[36] 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*, pages 2304–2314, 2019. 2

[37] Vincent Sitzmann, Semon Rezchikov, William T. Freeman, Joshua B. Tenenbaum, and Fredo Durand. Light field networks: Neural scene representations with single-evaluation rendering. In *Advances in Neural Information Processing Systems*, 2021. 1

[38] Shih-Yang Su, Frank Yu, Michael Zollhöfer, and Helge Rhodin. A-nerf: Articulated neural radiance fields for learning human shape, appearance, and pose. In *NeurIPS*, 2021. 1, 2, 3

[39] Ayush Tewari, Ohad Fried, Justus Thies, Vincent Sitzmann, Stephen Lombardi, Kalyan Sunkavalli, Ricardo Martin-Brualla, Tomas Simon, Jason Saragih, Matthias Nießner, et al. State of the art on neural rendering. In *Computer Graphics Forum*, volume 39, pages 701–727. Wiley Online Library, 2020. 1, 2

[40] Ayush Tewari, Justus Thies, Ben Mildenhall, Pratul Srinivasan, Edgar Tretschk, Wang Yifan, Christoph Lassner, Vincent Sitzmann, Ricardo Martin-Brualla, Stephen Lombardi, et al. Advances in neural rendering. In *Computer Graphics Forum*, volume 41, pages 703–735. Wiley Online Library, 2022. 1, 2

[41] Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of Machine Learning Research*, 9(86):2579–2605, 2008. 5, 7

[42] Urs Waldmann, Jannik Bamberger, Ole Johannsen, Oliver Deussen, and Bastian Goldlücke. Improving unsupervised label propagation for pose tracking and video object segmentation. In *DAGM German Conference on Pattern Recognition*, pages 230–245. Springer, 2022. 1

[43] Urs Waldmann, Alex Hoi Hang Chan, Hemal Naik, Máté Nagy, Iain D Couzin, Oliver Deussen, Bastian Goldluecke, and Fumihiro Kano. 3d-muppet: 3d multi-pigeon pose estimation and tracking. *arXiv preprint arXiv:2308.15316*, 2023. 1

[44] Urs Waldmann, Hemal Naik, Nagy Máté, Fumihiro Kano, Iain D Couzin, Oliver Deussen, and Bastian Goldlücke. I-muppet: Interactive multi-pigeon pose estimation and tracking. In *DAGM German Conference on Pattern Recognition*, pages 513–528. Springer, 2022. 1

[45] Tristan Walter and Iain D Couzin. Trex, a fast multi-animal tracking system with markerless identification, and 2d estimation of posture and visual fields. *eLife*, 10:e64000, 2021. 1

[46] He Wang, Srinath Sridhar, Jingwei Huang, Julien Valentin, Shuran Song, and Leonidas J Guibas. Normalized object coordinate space for category-level 6d object pose and size estimation. In *CVPR*, pages 2642–2651, 2019. 2, 3

[47] Jian Wang, Yunshan Zhong, Yachun Li, Chi Zhang, and Yichen Wei. Re-identification supervised texture generation. In *CVPR*, pages 11846–11856, 2019. 1, 2

[48] Shangzhe Wu, Ruining Li, Tomas Jakab, Christian Rupprecht, and Andrea Vedaldi. Magicpony: Learning articulated 3d animals in the wild. In *CVPR*, pages 8792–8802, 2023. 2

[49] Yiheng Xie, Towaki Takikawa, Shunsuke Saito, Or Litany, Shiqin Yan, Numair Khan, Federico Tombari, James Tompkin, Vincent Sitzmann, and Srinath Sridhar. Neural fields in visual computing and beyond. In *Computer Graphics Forum*, volume 41, pages 641–676. Wiley Online Library, 2022. 1, 2

[50] Alper Yilmaz, Omar Javed, and Mubarak Shah. Object tracking: A survey. *Acm computing surveys (CSUR)*, 38(4):13–es, 2006. 2

[51] Wentao Yuan, Zhaoyang Lv, Tanner Schmidt, and Steven Lovegrove. Star: Self-supervised tracking and reconstruction of rigid objects in motion with neural rendering. In *CVPR*, pages 13144–13152, 2021. 1, 2

[52] Zhedong Zheng, Liang Zheng, and Yi Yang. A discriminatively learned cnn embedding for person reidentification.

ACM transactions on multimedia computing, communications, and applications (TOMM), 14(1):1–20, 2017. 1, 2, 7

[53] Silvia Zuffi, Angjoo Kanazawa, Tanya Berger-Wolf, and Michael J. Black. Three-d safari: Learning to estimate zebra pose, shape, and texture from images "in the wild". In *ICCV*, October 2019. 2

[54] Silvia Zuffi, Angjoo Kanazawa, and Michael J. Black. Lions and tigers and bears: Capturing non-rigid, 3d, articulated shape from images. In *CVPR*, 2018. 2