# Neural Puppeteer: Keypoint-Based Neural Rendering of Dynamic Shapes
## Supplemental Material

Simon Giebenhain*, Urs Waldmann*[0000−0002−1626−9253], Ole Johannsen[0000−0002−7786−8516], and Bastian Goldluecke[0000−0003−3427−4029]

University of Konstanz, Konstanz, Germany
urs.waldmann@uni-konstanz.de
*Authors contributed equally.

**Abstract.** In the supplemental material, we provide further background information about our method as well as additional results. We start by giving a more detailed description of our network components and an in-depth discussion of our implementation details. Afterwards, we perform an ablation study on some of our design choices and discuss some of the trade-offs of our decisions. Finally, we showcase additional results on synthetic as well as real word examples. We also evaluate the expressiveness of our latent space by performing pose interpolation and visualize the choice of keypoints used for reconstruction. Please also check out our videos for further insights into the view consistency of our method, which you find on our project page https://urs-waldmann.github.io/NePu/.

# Table of Contents

1 Method Details . . . . . 3

1.1 Architectural Details and Hyperparameters . . . . . 3

Encoder . . . . . 3

Decoder . . . . . 3

Renderer . . . . . 4

Positional Embedding . . . . . 5

LFN Baseline . . . . . 5

1.2 Implementation Details . . . . . 5

1.3 Details on Inverse Rendering . . . . . 6

1.4 Details on LToHP Baseline . . . . . 6

2 Ablation Studies . . . . . 7

2.1 Hyperparameters and Architecture . . . . . 7

2.2 Pose Estimation by Inverse Rendering . . . . . 8

3 Additional Results . . . . . 9

3.1 Real World Examples . . . . . 9

3.2 Synthetic Examples . . . . . 11

3.3 Latent Space Interpolation . . . . . 15

4 Keypoint Distribution . . . . . 15

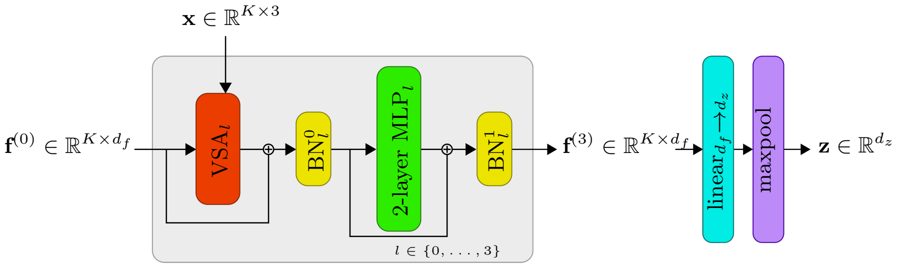

$\mathbf{x} \in \mathbb{R}^{K\times 3}$
$\mathbf{f}^{(0)} \in \mathbb{R}^{K\times d_f}$
$\text{VSA}_l$
$\text{BN}_l^0$
2-layer $\text{MLP}_l$
$\text{BN}_l^1$
$l \in \{0, \dots, 3\}$
$\mathbf{f}^{(3)} \in \mathbb{R}^{K\times d_f}$
$\text{linear}_{d_f \to d_z}$
maxpool
$\mathbf{z} \in \mathbb{R}^{d_z}$

Fig. 1: **Encoder architecture and hyperparameters.**

# 1 Method Details

## 1.1 Architectural Details and Hyperparameters

In this section we describe our model architecture in more detail and provide all model specific hyperparameters. Figures 1, 2 and 3 illustrate the architecture of the encoder, decoder and renderer, respectively, and are described in the following.

**Encoder** The encoder is composed of four vector self-attention layers (VSA) as introduced in [16], which alternate with BatchNorm [5] layers, skip connections and element-wise multi-layer perceptrons (MLPs) with two layers and one ReLU activation function. The design of this block is inspired by transformers [15]. The number of dimensions throughout all these layers is set to $d_f = 256$ for all experiments.

This stage retains translational equivariance, since VSA only uses relative positional information. Furthermore, we compute VSA over the complete graph, i.e. information is exchanged between all possible pairs of keypoints. Note that this is a very generic approach which does not take the piece-wise rigidity of the skeletal structure into account. Adopting a more specific network is a topic of possible future work. One simple solution would be to pairwise compute the positional difference $\mathbf{x}_{k_1} - \mathbf{x}_{k_2}$ between keypoints $k_1$ and $k_2$ in the local coordinate frame of $k_2$, in a similar fashion to [14].

The resulting intermediate features $\mathbf{f}^{(3)} \in \mathbb{R}^{K\times d_f}$ for each of the $K$ keypoints are subsequently projected to $d_z$-dimensional space using a single linear layer and finally reduced to $\mathbf{z} \in \mathbb{R}^{d_z}$ using max-pooling. For all experiments we use $d_z = 1024$. Note that because of this step the translation equivariance becomes invariance.

**Decoder** Compressing all relevant information into a single latent vector is one of the core competences of our model. This allows our model to learn a strong globally consistent prior, which is crucial for our inverse rendering approach and allows us to fit NePu to sparse observations, as briefly demonstrated in section 3.1.

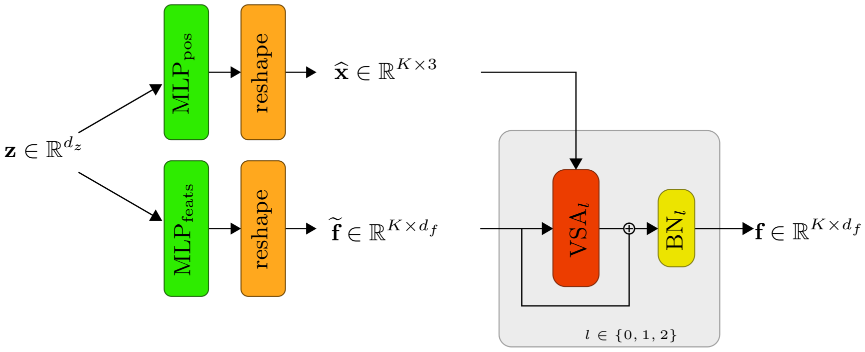

$\mathbf{z} \in \mathbb{R}^{d_z}$
$\mathrm{MLP}_{\mathrm{pos}}$
$\mathrm{MLP}_{\mathrm{feats}}$
reshape
reshape
$\hat{\mathbf{x}} \in \mathbb{R}^{K\times 3}$
$\tilde{\mathbf{f}} \in \mathbb{R}^{K\times d_f}$
$\mathrm{VSA}_l$
$\mathrm{BN}_l$
$\mathbf{f} \in \mathbb{R}^{K\times d_f}$
$l \in \{0,1,2\}$

Fig. 2: **Decoder architecture and hyperparameters.**

Contrary to related work that uses the global latent vector to condition a neural field [8,11,13], we aim to benefit from ideas of recent locally conditioned models [12,2,3]. Hence, a consequence of this compressed latent representation is the need for a decoder, that decompresses $\mathbf{z}$ into locally anchored information.

For this purpose, we reconstruct keypoints $\hat{\mathbf{x}}$ using a 3-layer MLP $\mathrm{MLP}_{\mathrm{pos}}$ with two ReLU activation functions. To tackle overfitting, we set the internal dimensionality of $\mathrm{MLP}_{\mathrm{pos}}$ to $d_{\mathrm{bottle}} = 128$. The output dimensionality is set to $K \cdot 3$, such that we can reshape to the desired shape.

Furthermore, we obtain latent vectors $\tilde{\mathbf{f}}$ using another 3-layer MLP $\mathrm{MLP}_{\mathrm{feats}}$. This time, we increase the dimensionality of the hidden layers to $2 \cdot d_z$ and $4 \cdot d_z$ respectively. The output dimension is $K \cdot d_f$.

We interpret $\tilde{\mathbf{f}}$ as local latent vector centered at the corresponding reconstruced keypoints $\hat{\mathbf{x}}$, similar to [3]. Finally, we refine $\tilde{\mathbf{f}}$ using three additional layers of VSA. Again, the internal dimensionality is $d_f$ and we use all $K$ keypoints as neighbors.

**Renderer** From a technical perspective, our renderer is very similar to [3], but has three prediction heads instead. Furthermore, we refine the features $\mathbf{f}$ a second time using another refinement block, after their positions $\hat{\mathbf{x}}$ have been projected to 2D.

The core of our renderer is the cross vector attention (CVA) module, as introduced in [3], which uses the queried pixel value $\mathbf{q} \in [0,1]^2$ and global latent vector $\mathbf{z}$ as query set and the local latent vector $\mathbf{f}_{2D}^{(2)}$ with their positional information $\mathbf{x}_{2D}^{*} \in [0,1]^2 \times \mathbb{R}$ as key-value set. Please note that the number of dimensions for $\mathbf{q}$ and $\mathbf{x}_{2D}^{*}$ do not match, due to the appended depth values. We circumvent this, by appending a zero to $\mathbf{q}$. In all experiments we use apply CVA to the 12 nearest neighbors of $\mathbf{q}$ in normalized pixel coordinates. Again, the internal dimensionality of CVA and the prediction heads is $d_f = 256$. For more details about the CVA module, we refer to [3].

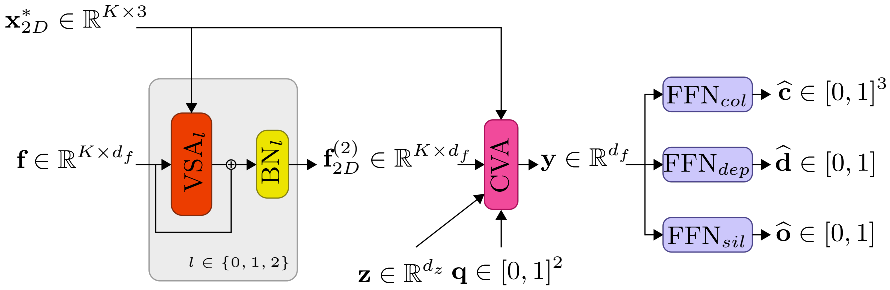

$\mathbf{x}^*_{2D} \in \mathbb{R}^{K\times 3}$
$\mathbf{f} \in \mathbb{R}^{K\times d_f}$
VSA$_l$
BN$_l$
$l \in \{0,1,2\}$
$\mathbf{f}^{(2)}_{2D} \in \mathbb{R}^{K\times d_f}$
CVA
$\mathbf{z} \in \mathbb{R}^{d_z}$
$\mathbf{q} \in [0,1]^2$
$\mathbf{y} \in \mathbb{R}^{d_f}$
FFN$_{col}$
FFN$_{dep}$
FFN$_{sil}$
$\hat{\mathbf{c}} \in [0,1]^3$
$\hat{\mathbf{d}} \in [0,1]$
$\hat{\mathbf{o}} \in [0,1]$

Fig. 3: **Renderer architecture and hyperparameters.**

**Positional Embedding** Compared to the design of the VSA and CVA modules in [16,3], we include a sinusoidal positional encoding, as proposed in [9]. To be more specific, we do not encode the positional difference between two keypoints $x_i$ and $x_j$ using a 2-layer MLP $\delta$ with one activation function

$$\mathrm{pos}_{x_i x_j} = \delta(\mathbf{x}_i - \mathbf{x}_j). \tag{1}$$

Instead, we use

$$\mathrm{pos}_{x_i x_j} = \delta(\sin(2^0\pi\mathbf{d}_{ij}), \cos(2^0\pi\mathbf{d}_{ij}), \dots, \sin(2^{F-1}\pi\mathbf{d}_{ij}), \cos(2^{F-1}\pi\mathbf{d}_{ij})), \tag{2}$$

where $\mathbf{d}_{ij} = \mathbf{x}_i - \mathbf{x}_j$ is the positional difference between $x_i$ and $x_j$ and the number of employed frequencies is $F = 5$. We ablate the effect of including this embedding in section 2.1.

**LFN Baseline** In the main paper, we compare NePu with a baseline that uses the *Light Field Network* (LFN) for rendering [13], which we denoted as LFN*. This baseline shares the encoder architecture with NePu, but lacks the decoder and has a much simpler rendering structure which predicts color values

$$\hat{c} = \mathrm{FFN}_{\mathrm{col}}(\mathbf{z}, \mathrm{pos}_q), \tag{3}$$

where the positional encoding $\mathrm{pos}_q$ is defined as in equation 2, but with $\mathbf{q}$ replacing $\mathbf{d}_{ij}$. Note that $\mathbf{q}$ is expressed in Plücker coordinates. Depth and occupancy values are predicted equivalently. To accommodate for the loss in expressiveness due to the global conditioning, we set the internal dimensionality of $\mathrm{FFN}_{\mathrm{col}}, \mathrm{FFN}_{\mathrm{dep}}$ and $\mathrm{FFN}_{\mathrm{sil}}$ to 512.

### 1.2 Implementation Details

**Additional Training Hyperparameters**. As mentioned in the main paper, we sample pixels uniformly in the ground truth silhouette to compute $\mathcal{L}_{\mathrm{dep}}$ and $\mathcal{L}_{\mathrm{col}}$ on. We sample 500 points for these two losses. For $\mathcal{L}_{\mathrm{sil}}$ we sample 1500 on the boundary of the silhouette and perturb these points using Gaussian noise. For half of the points we use a small standard deviation $\sigma_{\text{near}}$ and a larger value $\sigma_{far}$ for the other half. The specific values differ on the size of the animal with respect to the complete frame, as well as, the shape of the subject class. In particular we use $\sigma_{\text{near}} \in \{0.05, 0.015, 0.0125, 0.015\}$ and $\sigma_{\text{far}} \in \{0.175, 0.15, 0.125, 0.15\}$, where the values are listed in the order of humans, pigeons, cows and giraffes.

For the LFN-based baseline we observed, that the model was not capable of representing the silhouettes with enough detail, resulting in a disproportionate amount of emphasis on $\mathcal{L}_{\text{sil}}$. Hence, we eased the task by changing the sampling strategy for $\mathcal{L}_{\text{sil}}$ to uniform sampling as done in [8], where half of the points are sampled inside and half outside of the ground truth mask.

Additionally, we use gradient clipping with a threshold of 1.0 for all models.

**Data Augmentation**.

We employ two types of data augmentation. First we added Gaussian noise to the keypoints. Depending on the object class we used a standard deviation $\sigma \in \{0.001, 0.0005, 0.003, 0.003\}$ for humans, pigeons, cows and giraffes, respectively. Second, we randomly rotate the keypoints by a multiple of 45° degrees around the $z$-axis and shift the camera order accordingly. This way the models have better chances to learn rotation equivariance.

**Training, validation and test splits**. We split our animations for each animal in chunks of 10 time steps and randomly sample from those chunks a training, validation and test split of 70%/10%/20%. Our synthetic human data set contains distinct poses with no temporal connection. All samples are split randomly to obtain a training, validation and test split of 70%/10%/20%.

### 1.3 Details on Inverse Rendering

To solve the inverse rendering based keypoint estimation, we use the limited memory BFGS [10] (LM-BFGS) optimizer, when no prior information is known. We conduct 10 steps of LM-BFGS, which takes around 14 seconds. As a comparison when using the Adam optimizer [7], convergence sometimes takes up to 1000 steps, which amounts to 3 minutes of runtime.

The clustering procedure described in the main paper usually results in between 20 and 25 clusters for all animal datasets and in up to 40 clusters for the human dataset. Since rotations around the $z$-axis can impose problems for the optimization (e.g. starting from an unfavourable rotation, the optimization can easily get stuck in a local minima, where the head and tail of a pigeon are switched) we additionally rotate each cluster in 90° degree steps.

Furthermore, we sample 10000 pixels for each of the eight camera views, where half is sampled uniformly inside the observed silhouette and half outside. Note that this differs to our training procedure, but leads to more stable optimization.

### 1.4 Details on LToHP Baseline

In their algebraic triangulation solution, the authors of [6] predict 2D keypoints with a 2D backbone for every camera view independently. Then they learn an

Table 1: *Ablation study: Influence of the refinement, depth and sinosoidal positional embedding block on the losses.* Best result per column is bold. The configuration that we use in our paper is highlighted in gray.

| | Keyp. Loss:<br>MPJPE [mm] | Depth Loss:<br>MAE [mm] | Color Loss:<br>PSNR [dB] | Mask Loss:<br>IoU [%] |
|---|---|---|---|---|
| w/o ref., w/o depth | **19** | 32.7 | 18.53 | 97.88 |
| w/o ref., w/ depth | 24 | 26.2 | 18.53 | 97.86 |
| w/ ref. w/ depth | 24 | **24.1** | **18.75** | **98.09** |
| w/o pos. emb. | 20 | 30.7 | 16.91 | 94.28 |

Table 2: *Ablation study: Influence of the dimension $d_z$ of the global representation* $\mathbf{z}$ *on the losses.* Best result per column is bold. The configuration that we use in our paper is highlighted in gray.

| Dim.<br>$d_z$ | Keyp. Loss:<br>MPJPE [mm] | Depth Loss:<br>MAE [mm] | Color Loss:<br>PSNR [dB] | Mask Loss:<br>IoU [%] |
|---|---|---|---|---|
| 256 | 74 | 26.2 | 18.56 | 98.06 |
| 512 | 55 | 25.4 | 18.60 | 98.08 |
| 1024 | 24 | **24.1** | **18.75** | **98.09** |
| 2048 | **19** | 24.6 | 18.67 | 98.11 |

algebraic triangulation where they fuse all the 2D keypoints with the keypoints' confidences from the 2D backbone. In their volumetric triangulation solution, the authors predict 2D features for every camera view independently. Then they unproject the 2D features into a cuboid volume and pass it into a 3D convolutional neural network. The cuboid volume is set around a root joint, the pelvis.

For the volumetric model, we use either our predicted pelvis of the algebraic solution or the ground truth pelvis because for some objects in our data set, their algebraic model does not converge. We train both, the algebraic and volumetric model, for 500 epochs.

## 2 Ablation Studies

We perform several ablation studies with our cow data set.

### 2.1 Hyperparameters and Architecture

We perform an ablation study on the influence of the presence and absence of our refinement, depth and sinusoidal positional embedding blocks. Results are presented in Table 1. We see that all three choices are essential for our framework.

We also perform an ablation study on the influence of the dimension $d_z$ of our global representation $\mathbf{z}$. Results are in Table 2. We see that $d_z = 1024$ is the best choice for all losses.

Table 3: *Ablation study: Influence of the number of nearest neighbours on the losses.* Best result per column is bold. The configuration that we use in our paper is highlighted in gray.

| Number<br>of NN | Keyp. Loss:<br>MPJPE [mm] | Depth Loss:<br>MAE [mm] | Color Loss:<br>PSNR [dB] | Mask Loss:<br>IoU [%] |
|---|---|---|---|---|
| 2 | 30 | 29.6 | 17.70 | 97.78 |
| 4 | 27 | 27.0 | 18.17 | 98.02 |
| 8 | **24** | 24.7 | 18.69 | 98.07 |
| 12 | **24** | 24.1 | **18.75** | **98.09** |
| 16 | 31 | 23.8 | 18.63 | 98.06 |
| 25 (all) | 28 | **23.6** | 18.65 | 98.07 |

Table 4: *Ablation study: Influence of the number of cameras on the keypoint estimation.* Best result per column is bold. The configuration that we use in our paper is highlighted in gray.

|  | 1 cam | 2 cams | 4 cams | 8 cams |
|---|---|---|---|---|
| MPJPE [mm] | 143 | 49 | 19 | **15** |

Additionally we perform an ablation study on the influence of the number of nearest neighbours. Results are in Table 3. We see that 12 nearest neighbours is the best choice.

### 2.2 Pose Estimation by Inverse Rendering

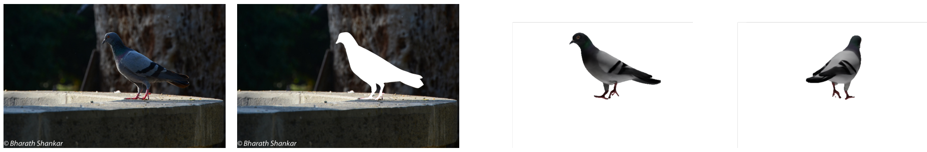
We perform an ablation study on the influence of the number of cameras and number of sample points on our keypoint estimation. We restrict this ablation to a maximum of eight cams since this is the number of cameras we can fit on a GPU with 40GB for our baseline [6]. Results are in Table 4 where we sampled 10000 points. We see that the performance of our model improves with more cameras. Furthermore it seems that a higher number of sample points leads to a better result: 15mm (10000 sample points, cf. Tab. 4 and Table 2 in paper) vs. 32mm (only 1000 sample points).

# 3 Additional Results

## 3.1 Real World Examples

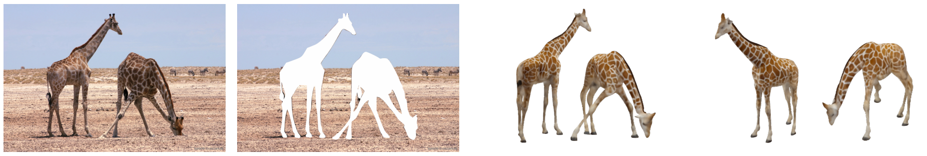
We conduct a small study on real world examples to demonstrate the flexibility of our approach, which arises from silhouette-based model fitting and the consistency provided by the global latent space. For this purpose, we fit our model to the silhouette on a single image and render novel views from it, as illustrated in Figs. 4 to 7.

More specifically, we choose one real world image for each subject class. Subsequently, we annotate the silhouette by hand and align the silhouette with our training setup. This includes centering the silhouette from left to right and aligning the floor with the training data. Furthermore, we scale the hand-annotated silhouette to the size of subject that our model learned. Please note that we only fit on silhouette data, thus the reconstructions show a different texture than the input image.



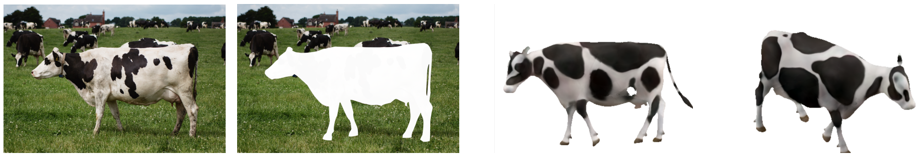
(a) Input View (b) Given Mask (c) Reconstruction (d) Novel view

Fig. 4: **Real world example: Pigeon.** Reconstruction of a Pigeon from a single pose.



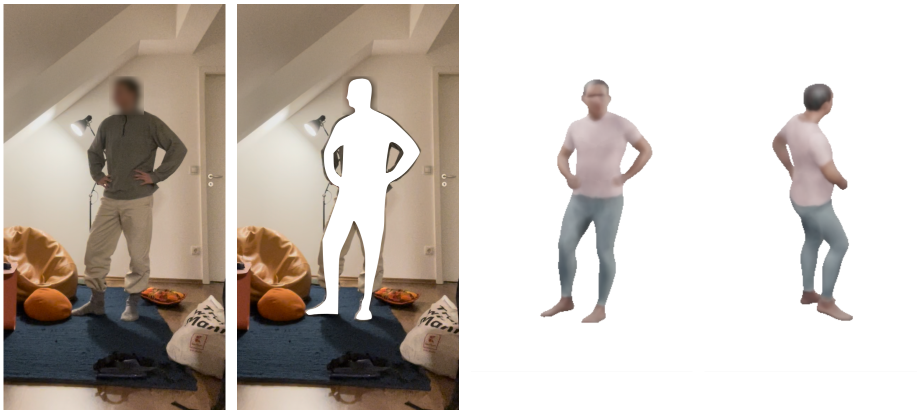
(a) Input View (b) Given Mask (c) Reconstruction (d) Novel view

Fig. 5: **Real world example: Giraffe.** We take a real world image of giraffes, generate masks for both and perform pose estimation using the silhouettes. The view to the right shows both giraffes from a different perspective. Please note that both giraffes were reconstructed and rendered individually. As our input data only includes data of a drinking giraffe with its hind legs closed, the reconstruction also has closed legs.



(a) Input View (b) Given Mask (c) Reconstruction (d) Novel view

Fig. 6: **Real world example: Cow.** Reconstruction of a cow from a single silhouette.



(a) Input View (b) Given Mask (c) Reconstruction (d) Novel view

Fig. 7: **Real world example: Human.** The pose estimation works well for a single silhouette of a human subject. However, ambiguities in the silhouette lead to a wrong rotation of the head. Multi-view data would help here with finding for example a correct nose position.

### 3.2 Synthetic Examples

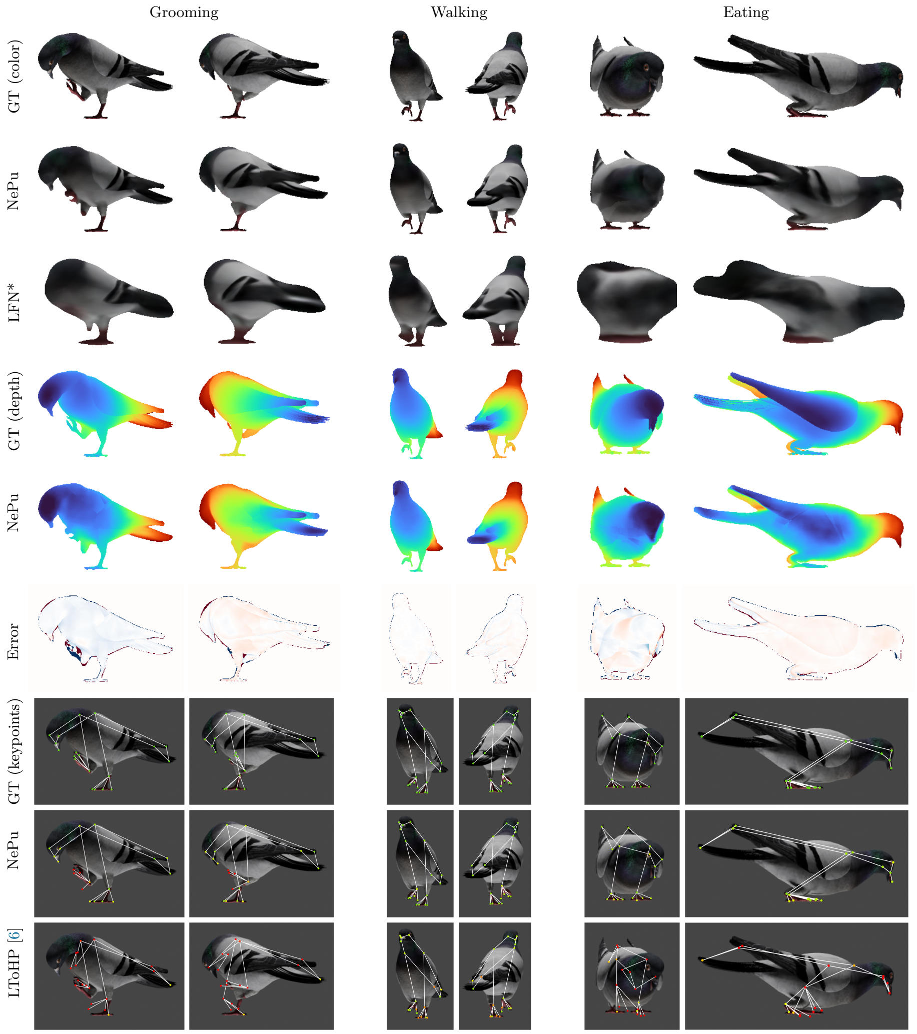


Fig. 8: **Additional Results: Pigeon.** The results for the pigeon are very satisfactory overall. However, some details around the legs are lost as we did not include a keypoint at the root of the claw. Thus, for example for the first pose, the rotation of the hand is reconstructed differently.

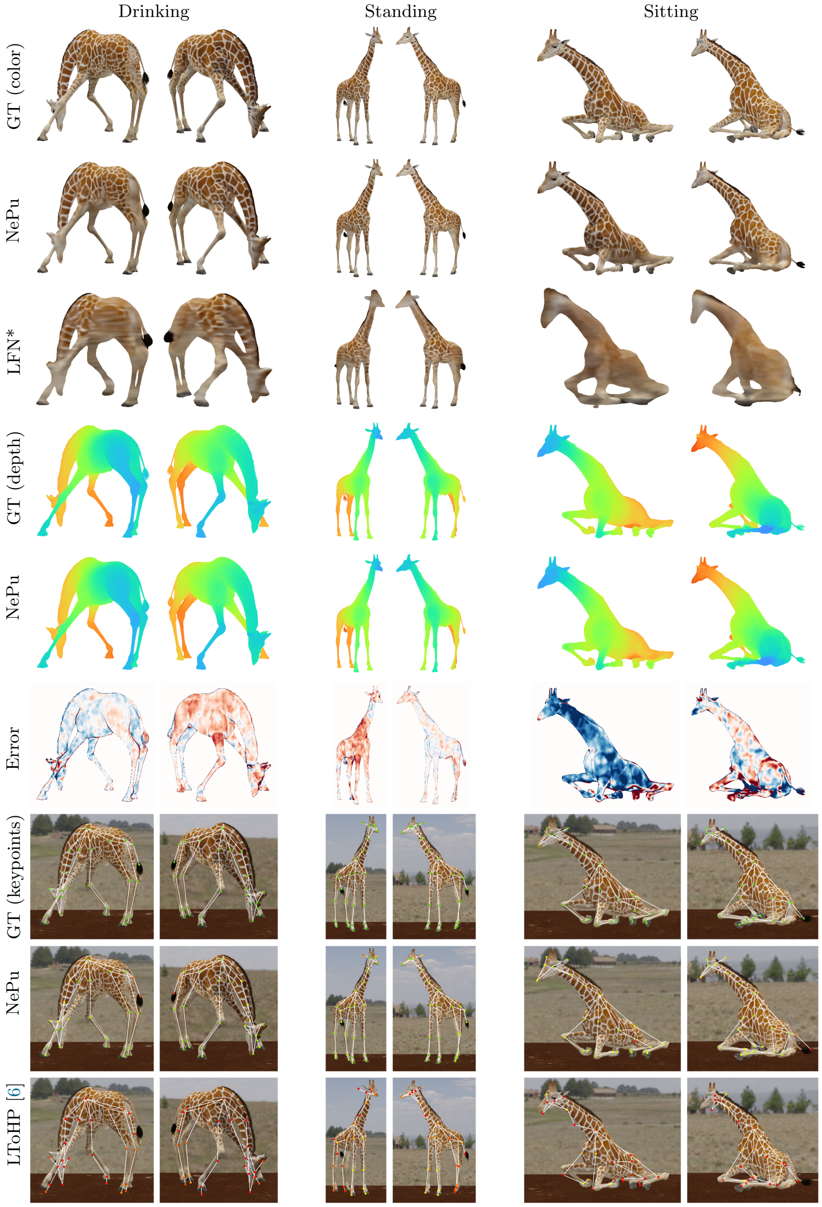


Fig. 9: **Additional Results: Giraffe.** The reconstruction of the giraffe works very well overall. The details of the skin are well reconstructed up to a certain frequency, with some oversmoothing occuring for high-frequency texture content, such as at the legs. However, the quality of the results is overall substantially better to the baseline version $LFN^*$ without local conditioning.

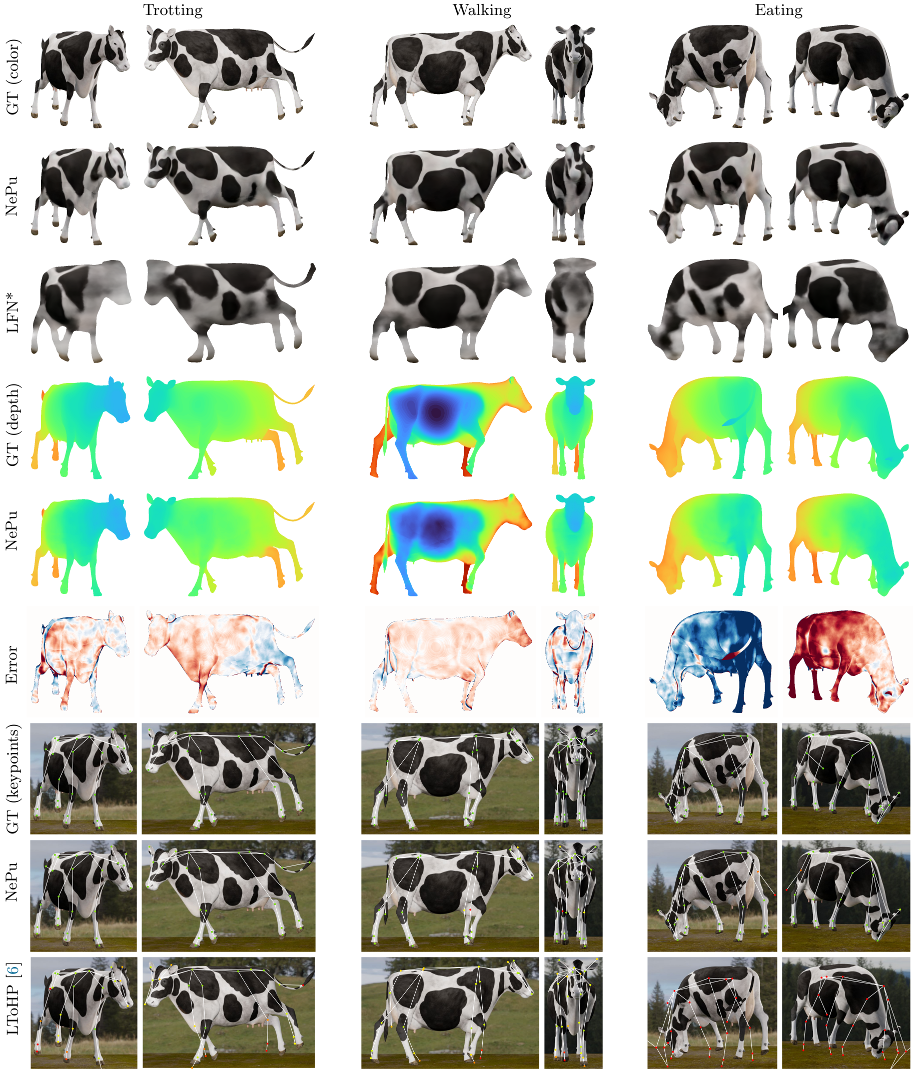


Fig. 10: **Additional Results: Cow.** The additional results on the cow dataset show the importance of the frequency of keypoints. For example, two keypoints on the tail are most likely necessary to reliably reconstruct the curvature.

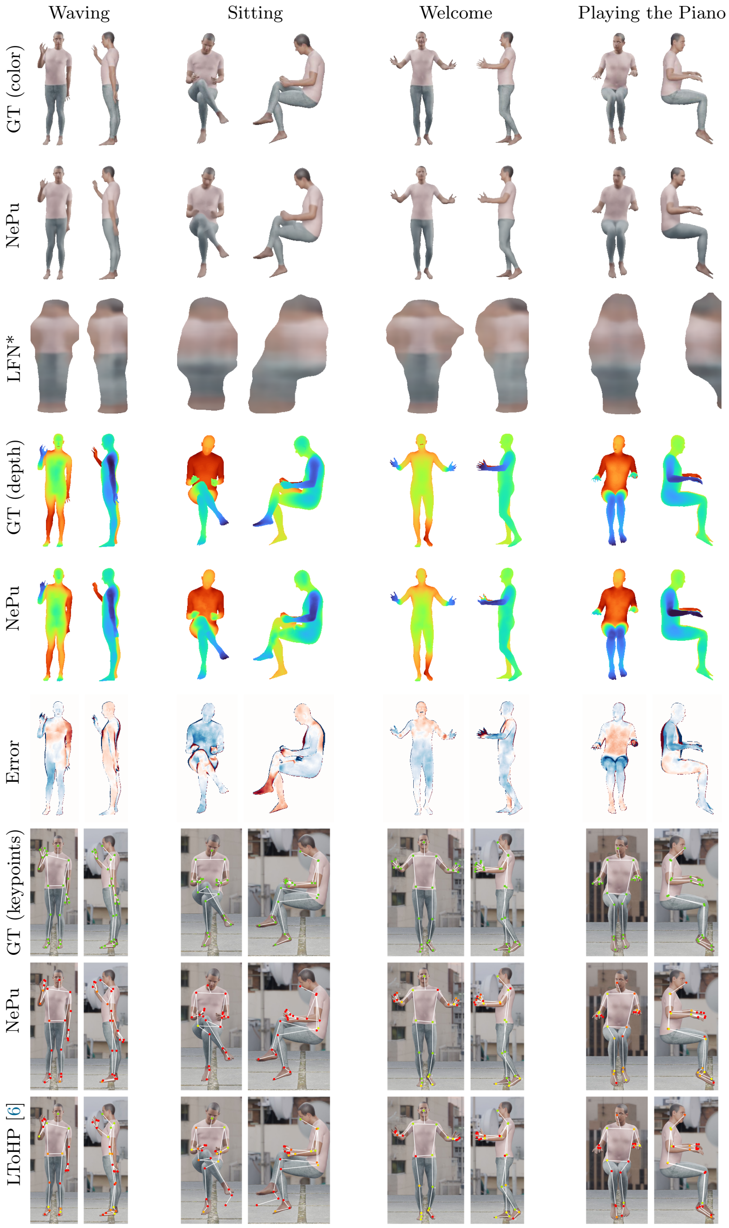


Fig. 11: **Additional Results: Human.** We show four additional poses which the network has never seen before. Our method performs well on poses similar to ones seen in the training data, but runs into problems if ambiguities are present from the silhouettes. For example, in *Sitting* the left and right leg are swapped as they can hardly be distinguished from the silhouette alone .

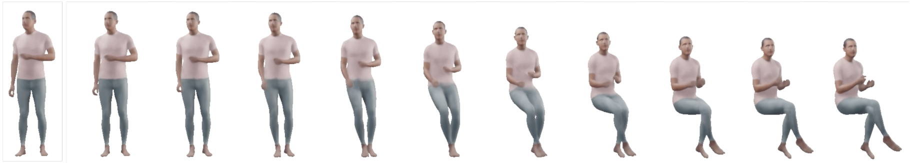


Fig. 12: **Latent Space Interpolation.** The structure of the latent space encodes meaningful information about how individual poses are related to each other. We show this by interpolating between two known poses (far left and right) in latent space. The interpolation yields a realistic motion of the individual joints.

### 3.3 Latent Space Interpolation

We demonstrate the expressiveness of the pose latent space, $\mathbf{z} \in \mathbb{R}^{d_z}$ by evaluating the linear interpolation between two known poses in this space, and rendering from the interpolated representation. The results can be seen in Fig. 12. More results are demonstrated in the supplementary material.

## 4 Keypoint Distribution

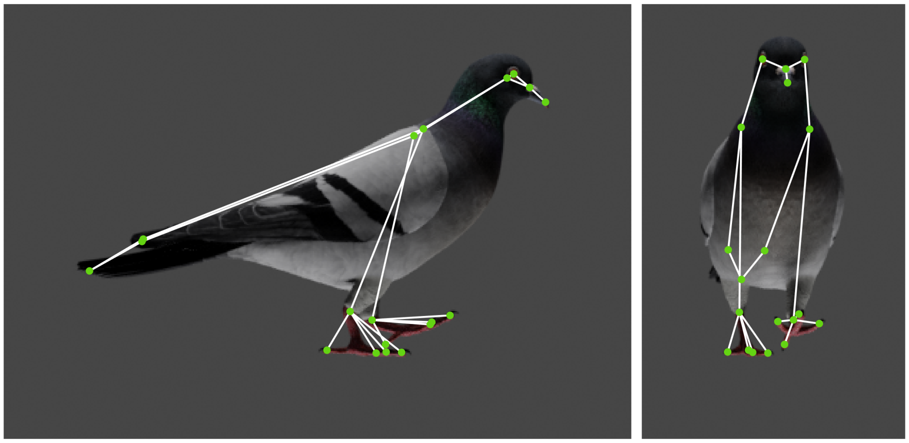


Fig. 13: **Keypoint Locations: Pigeon.** The model of the pigeon includes a total of 19 keypoints. The following keypoints are included: two each per eye, shoulder, wing tip, knee, four per body side for the tips of all claws (3 forward facing, one backward facing), and one each for tip of beak, root of nose, and tip of tail feathers.

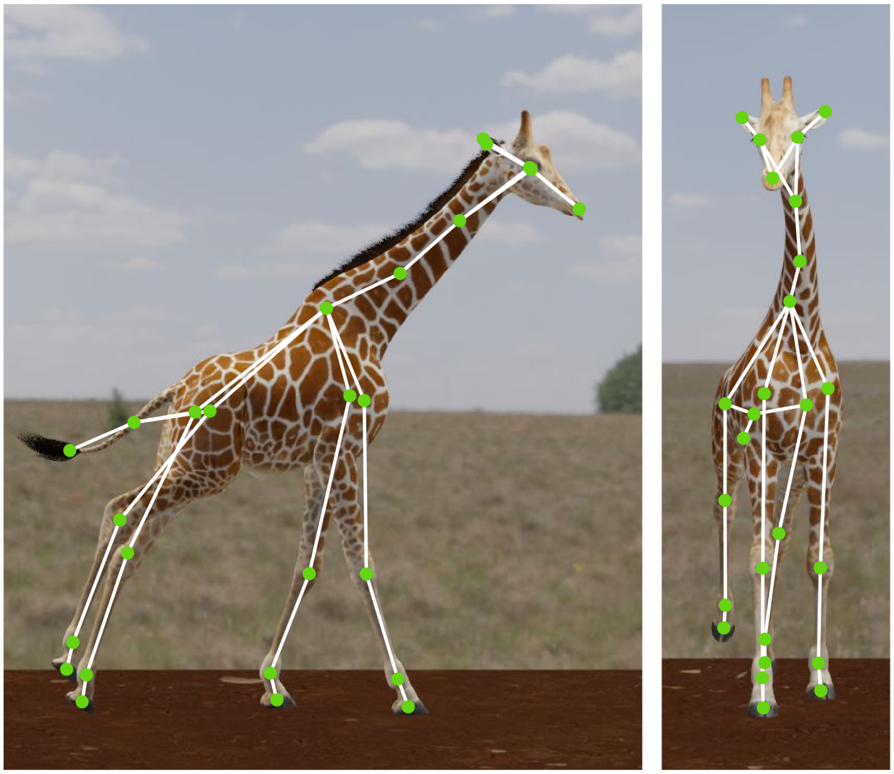


Fig. 14: **Keypoint Locations: Giraffe.** The model of the giraffe includes a total of 26 keypoints. The following keypoints are included: two each for eyes, ears, shoulders, hips, four each for knees, heels, and hooves, one each for nose, upper neck, lower neck, hump, middle tail and lower tail.

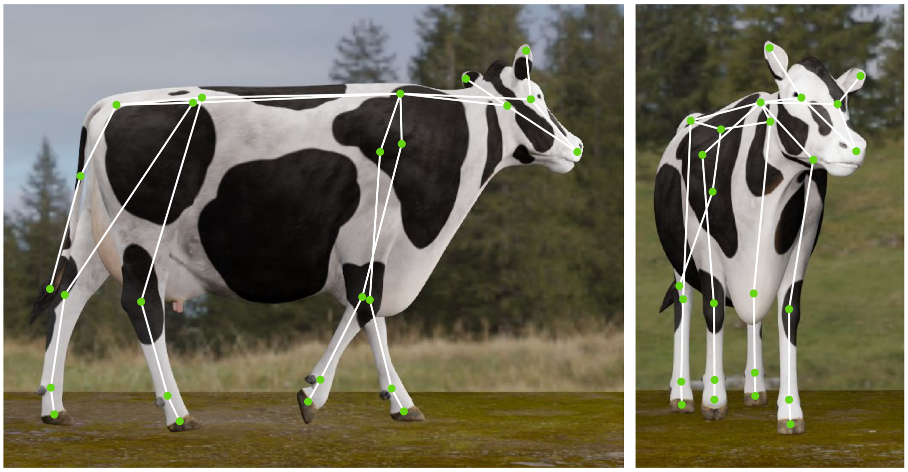


Fig. 15: **Keypoint Locations: Cow.** The model of the cow includes a total of 25 keypoints. The following keypoints are included: two each for eyes, ears, shoulders, hips, four each for knees, heels, and hooves, plus one each for nose, hump, back, middle tail and lower tail.

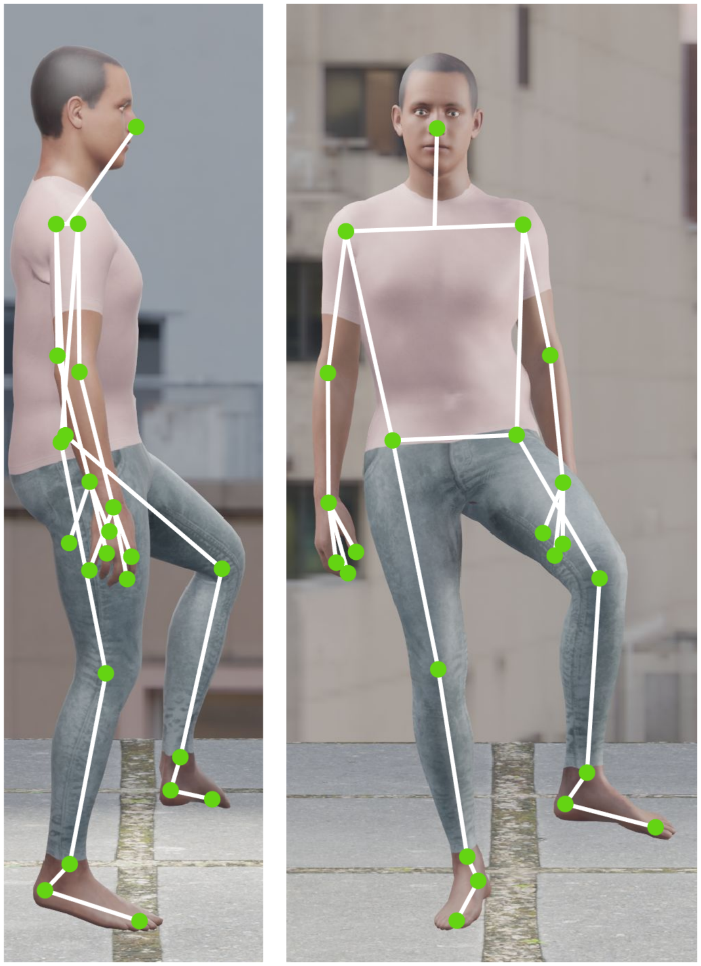


Fig. 16: **Keypoint Locations: Human.** The model of the human includes the same 33 keypoints as [1,4], but for the purpose of better visualization, the only keypoint in the face we show is the nose. The following keypoints are included: three per eye, one for each ear, corner of the mouth, shoulder, elbow, wrist, pinky, index finger, thumb, hip, knee, ankle, heel, foot, plus the central keypoint for the tip of the Nose.

## References

1. Bazarevsky, V., Grishchenko, I., Raveendran, K., Zhu, T., Zhang, F., Grundmann, M.: Blazepose: On-device real-time body pose tracking. arXiv preprint arXiv:2006.10204 (2020) 18
2. Chibane, J., Alldieck, T., Pons-Moll, G.: Implicit functions in feature space for 3d shape reconstruction and completion. In: CVPR (2020) 4
3. Giebenhain, S., Goldlücke, B.: Air-nets: An attention-based framework for locally conditioned implicit representations. In: 2021 International Conference on 3D Vision (3DV). pp. 1054–1064 (2021) 4, 5
4. Google: Google ai blog (2020), https://ai.googleblog.com/2020/08/on-device-real-time-body-pose-tracking.html 18
5. Ioffe, S., Szegedy, C.: Batch normalization: Accelerating deep network training by reducing internal covariate shift. In: Proceedings of the 32nd International Conference on Machine Learning (ICML). Proceedings of Machine Learning Research, vol. 37 (2015) 3
6. Iskakov, K., Burkov, E., Lempitsky, V., Malkov, Y.: Learnable triangulation of human pose. In: ICCV (2019) 6, 8, 11, 12, 13, 14
7. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. In: 3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings (2015), http://arxiv.org/abs/1412.6980 6
8. Mescheder, L., Oechsle, M., Niemeyer, M., Nowozin, S., Geiger, A.: Occupancy networks: Learning 3d reconstruction in function space. In: CVPR (2019) 4, 6
9. Mildenhall, B., Srinivasan, P.P., Tancik, M., Barron, J.T., Ramamoorthi, R., Ng, R.: Nerf: Representing scenes as neural radiance fields for view synthesis. In: European Conference on Computer Vision (ECCV) (2020) 5
10. Nocedal, J., Wright, S.J.: Numerical Optimization. Springer, New York, NY, USA, 2e edn. (2006) 6
11. Park, J.J., Florence, P., Straub, J., Newcombe, R., Lovegrove, S.: Deepsdf: Learning continuous signed distance functions for shape representation. In: CVPR (2019) 4
12. Peng, S., Niemeyer, M., Mescheder, L., Pollefeys, M., Geiger, A.: Convolutional occupancy networks. In: ECCV (2020) 4
13. Sitzmann, V., Rezchikov, S., Freeman, W.T., Tenenbaum, J.B., Durand, F.: Light field networks: Neural scene representations with single-evaluation rendering. In: Advances in Neural Information Processing Systems (2021), https://openreview.net/forum?id=q0h6av9Vi8 4, 5
14. Su, S.Y., Yu, F., Zollhöfer, M., Rhodin, H.: A-nerf: Articulated neural radiance fields for learning human shape, appearance, and pose. In: NeurIPS (2021) 3
15. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, Ł., Polosukhin, I.: Attention is all you need. NeurIPS **30** (2017) 3
16. Zhao, H., Jiang, L., Jia, J., Torr, P., Koltun, V.: Point transformer. In: International Conference on Computer Vision (ICCV) (2021) 3, 5