Learning Neural Volumetric Representations of Dynamic Humans in Minutes Supplemental Material

Chen Geng* Sida Peng* Zhen Xu* Hujun Bao Xiaowei Zhou†

State Key Laboratory of CAD&CG, Zhejiang University

Abstract

This supplementary material has the following contents.

1. *Sec. A presents the implementation details, including the derivation of transformation matrices (Sec. A.1), the computation of blend weights and UV coordinates (Sec. A.2), network architectures (Sec. A.3), hyperparameters (Sec. A.4), loss terms (Sec. A.5), and the body decomposition (Sec. A.6).*
2. *Sec. B describes the details of evaluation and experiments.*
3. *Sec. C presents the details of baseline methods.*
4. *Sec. D shows additional experimental results and ablation studies.*

A. Implementation details

A.1. Derivation of transformation matrices

Assuming there are K human bones, [5] defines the human skeleton as $(\mathbf{J}, \theta)$, in which bone position is represented by $\mathbf{J} \in \mathbb{R}^{3\times K}$ and bone rotation is represented by $\theta \in \mathbb{R}^{3\times(K+1)} = [\omega_0^T, \omega_1^T, \cdots, \omega_K^T]$. Given the target pose θ_t, the transformation G_k of the k-th bone is formulated as:

$$G_k = A_k(\mathbf{J}, \theta_t) A_k(\mathbf{J}, \theta_c)^{-1}, \quad (1)$$

$$A_k(\mathbf{J}, \theta) = \prod_{i \in P(k)} \begin{bmatrix} R(\omega_i) & \mathbf{j}_i \\ 0 & 1 \end{bmatrix}, \quad (2)$$

where θ_c is the canonical pose, $R(\omega_i) \in \mathbb{R}^{3\times 3}$ denotes the rotation matrix of ω_i, $\mathbf{j}_i$ denotes the i-th joint location, and $P(k)$ denotes the ordered set of parent joints of joint k.

*Equal contribution

A.2. Computation of blend weights and UV coordinates

For a query point $\mathbf{x}$, we compute its blend weights and UV coordinates from the SMPL model as follows. First, we find the nearest surface point on the SMPL mesh. Then, the corresponding mesh facet gives three mesh vertices. The blend weights and UV coordinates of mesh vertices are pre-defined by the SMPL model. Finally, we perform barycentric interpolation to compute the blend weights and UV coordinates of the query point.

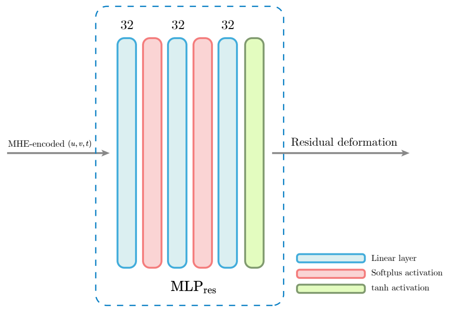
A.3. Network architectures

Figure 1a and 1b illustrate the network architecture of the residual deformation module and our proposed part representation.

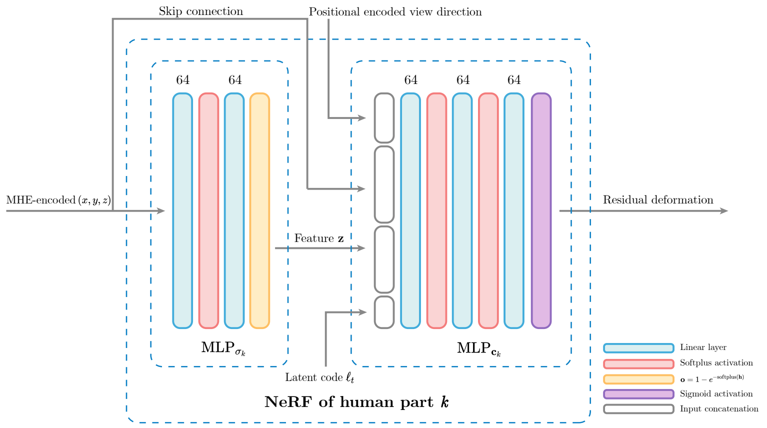
For the residual deformation module, we use an MLP with 3 layers of width 32 and Softplus activation for the 2 hidden layers. We apply $\tanh$ on the deformation output. The residual deformation module takes as input the MHE-encoded (u, v, t) coordinates, and outputs 3D translational vectors.

For the canonical human model, we use an MHE-augmented NeRF network for every part of our part representation. The NeRF network of each part is comprised of two MLPs, with 2 and 3 layers of width 64 respectively for volume density and color. Specifically, MHE-encoded (x, y, z) coordinates in canonical space are first fed into the first MLP to output a 16D geometric feature $\mathbf{z}$, which is concatenated with positional encoded [6] canonical view direction, $\mathbf{z}$ and an 8D time-varying latent code [9]. Then, the concatenated vector is passed into the second MLP. We apply Softplus activation on all hidden layers of the two MLPs and a Sigmoid activation for the 3D color output. Following [8], the first MLP outputs occupancy directly instead of volume density, and we later use the occupancy as the alpha value during volume rendering. The occupancy output $\mathbf{h}$ is activated using $\mathbf{o} = 1 - e^{-\mathrm{softplus}(\mathbf{h})}$. The volume rendering process is implemented using alpha composition following [8].

Both of the previous two modules use the same MHE



(a) NeRF Network architecture for residual deformation field $\mathrm{MLP}_{\mathrm{res}}$.



(b) NeRF Network architecture for every part of the canonical human MLP_{σ_k} and $\mathrm{MLP}_{\mathbf{c}_k}$.

	Body	Head	Left Arm	Right Arm	Leg
Number of Levels	16	16	16	16	16
Hash Table Size	2^{20}	2^{18}	2^{15}	2^{15}	2^{20}
Feature Dimension	16	16	16	16	16
Coarsest Resolution	16	2	2	2	2
Finest Resolution	2768	346	346	346	346

Table 1. Hyperparameters for Multiresolution Hash Table of Canonical Part Representation.

implementation. We follow [7] to use two separate arrays to store encodings of lower resolution dense grid and another array for the bigger arrays requiring hash encodings, as well as the indexing and spacial hashing algorithm. We choose $19349663, 83492791$ as the two primes π_2, π_3 described in [7]. We concatenate the hash-encoded coordinates with the original coordinates as input to the next module. The detailed parameters of the MHE for different parts of the canonical human body and the residual deformation will be explained in A.4.

A.4. Hyperparameters

For the part-based representation of the canonical human model, the default configuration of multi-resolution hash encoding shared by all experiments is shown in Table 1. We then sum the features from all levels and feed them into the MLP shown in Figure 1b to get density and color.

	UVT-Deformation
Number of Levels	8
Hash Table Size	2^{14}
Feature Dimension	2
Coarsest Resolution	4
Finest Resolution	53

Table 2. Hyperparameters for Multiresolution Hash Table for UVT-Deformation Field.

For the motion field, the configuration of (u, v, t)-based multi-resolution hash encoding shared by all experiments is shown in Table 2. The features are then concatenated and fed into MLP shown in Figure 1a to get the non-rigid residual output.

A.5. Loss terms

In addition to the rendering loss L_{rgb}, we introduce a loss L_{dist} to enforce the density field to concatenate on the surface [1], and two losses L_{res} and L_{smooth} to ensure the predicted residual deformations are small and smooth. The regularization L_{dist} on the density field [1] is defined as:

$$L_{\mathrm{dist}} = \sum_{i,j} w_i w_j \left| \frac{z_i + z_{i+1}}{2} - \frac{z_j + z_{j+1}}{2} \right|, \tag{3}$$

where w_i means the weight of i-th sampled point on each ray and z_i means the depth of this point. The regularization terms L_{res} and L_{smooth} on the residual deformation field are defined as:

$$L_{\mathrm{res}} = ||\mathrm{MLP}_{\mathrm{res}}(\psi_{\mathrm{res}}(u_k, v_k, t))||_2, \tag{4}$$

$$L_{\mathrm{smooth}} = ||\mathrm{MLP}_{\mathrm{res}}(\psi_{\mathrm{res}}(u'_k, v'_k, t)) - \mathrm{MLP}_{\mathrm{res}}(\psi_{\mathrm{res}}(u''_k, v''_k, t))||_2, \tag{5}$$

where (u'_k, v'_k) denotes the UV coordinate of sampled point $\mathbf{x}'$, and u''_k, v''_k are derived from points $\mathbf{x}''$ sampled within a 0.01 radius of $\mathbf{x}'$.

The total loss for training is defined as:

$$L = \lambda_1 L_{\text{rgb}} + \lambda_2 L_{\text{res}} + \lambda_3 L_{\text{dist}} + \lambda_4 L_{\text{smooth}}. \quad (6)$$

In all experiments, we choose $\lambda_1 = 1$, $\lambda_2 = 0.1$, $\lambda_3 = 0.1$, and $\lambda_4 = 1.0 \times 10^{-4}$.

A.6. Body decomposition

We use the blend weights of SMPL model [5] to decompose the parametric model into multiple parts. For each mesh vertex, we first find the bone that has the biggest blend weight and then convert the corresponding bone to the part using Table 3.

body	0, 3, 6, 9, 13, 14
leg	1, 2, 4, 5, 7, 8, 10, 11
head	12, 15
left arm	16, 18, 20, 22
right arm	17, 19, 21, 23

Table 3. Conversion from bones to parts.

The decomposed human body is shown in Fig. 2.

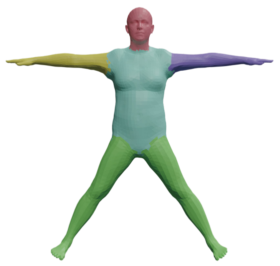


Figure 2. The part decomposition of canonical human pose.

A.7. Other implementation details

The whole framework is written in PyTorch, and no customized CUDA kernel is written for a fair comparison. We use PyTorch3D [12] framework to find the closest point on the part from sampled points.

When sampling patches, we use provided human segmentation mask to guide the sampling to ensure most of the patches present information about the performer.

To predict the density and color of a query point $\mathbf{x}$, we first find the nearest surface point $\mathbf{p}_k$ on each human part. When the distance $d_k = \|\mathbf{x} - \mathbf{p}_k\|$ is bigger than a threshold τ, we simply set the density σ_k and color $\mathbf{c}_k$ as zero for the k-th part. Otherwise, we pass the query point through the networks to regress the density and color. In all experiments, we set τ as 0.1.

B. Datasets and metrics

ZJU-MoCap This dataset is a public dataset for non-commercial research purposes. Any other use is prohibited. One need to sign an agreement to use this dataset. We choose 377, 386, 387, 392, 393 and 394 for evaluation. 313 and 315 are not chosen because they only show one side of human body in video from only single view. All sequences use camera id 4 for input. We use 100 frames for training and sample 1 frame every 5 frames.

MonoCap This dataset consists of two videos ”Lan” and ”Marc” from DeepCap dataset [3] and two videos ”Olek” and ”Vlad” in DynaCap [2]. The detailed configuration for this dataset and ZJU-MoCap is shown in Table 4.

The DeepCap dataset [3] and DynaCap dataset [2] are only granted for non-commercial academic purposes. They prohibit the redistribution of that data. The users should also sign a license.

Note that both the ZJU-Mocap and MonoCap datasets do not contain any personally identifiable information or offensive content.

Metrics For each data, we evaluate PSNR, SSIM and LPIPS* to compare our method and baselines. We compute the metrics by comparing the whole rendered image and the ground-truth image. The SSIM metric is evaluated using Scikit-image [13], and LPIPS metric is evaluated using [16]. We report LPIPS* for more clear comparison and LPIPS* = LPIPS $\times 10^3$.

C. Details of baselines

Neural Body [11], Ani-SDF [10], HumanNeRF [14] and Ani-NeRF [9] We use the released code and conduct experiments on a single NVIDIA RTX 3090 GPU.

Neural Human Performer [4] and PixelNeRF [15] We use the released code for evaluation. The evaluation of these two generalizable methods is split into two stages: pretraining and finetuning. For pretraining, we conduct the training on the different datasets, as described in the main paper. We use all possible views and frames in the dataset for training and select one random view as a reference view. The whole pretrain stage lasts for about 10 hours to converge. For finetuning, we finetune the model on provided training views. Specifically, as the experiments conducted in the paper only provide one single view, thus these models can only be trained on this single view for fair comparison. They are also trained and finetuned on a single NVIDIA RTX 3090 GPU.

We provide the results of these two methods without finetuning on ”377” sequence in Table 5

	Training Views	Test Views	Start Frame	End Frame	Frame Interval
ZJU-MoCap	“4”	Remaining	0	500	5
MonoCap - Lan	“0”	Remaining	620	1120	5
MonoCap - Marc	“0”	Remaining	35000	35500	5
MonoCap - Olek	“44”	“0”, “5”, “10”, “15”, “20”, “25”, “30”, “35”, “40”, “45”, “49”	12300	12800	5
MonoCap - Vlad	“66”	“0”, “10”, “20”, “30”, “40”, “50”, “60”, “70”, “80”, “90”, “100”	15275	15775	5

Table 4. Training and test data of ZJU-MoCap and MonoCap datasets.

	PSNR	SSIM	LPIPS*
NHP [4] (No Finetuning)	24.22	0.946	60.9
PixelNeRF [15] (No Finetuning)	23.28	0.898	121.3

Table 5. Result of two generalizable baselines without finetuning, on ”377” dataset

D. Experiments

D.1. Supplementary results

We provide complete quantitative comparison results of ZJU-MoCap and MonoCap in Table 6 and Table 7 respectively.

We provide more qualitative results in the supplementary video.

D.2. Ablation on training views

Most view synthesis methods give better results when they are provided more input information. In this section, we present the comparison of our method and baseline methods when there are 4 views provided for training. We use the ”377” sequence from ZJU-MoCap to conduct experiments. We choose ”0”, ”6”, ”12” and ”18” views for training and remaining views for testing.

The results is shown in Table 8. Most methods behave higher performance when the number of input views increases. However, our method still show competitive result using a significantly faster training speed. Our method converges slightly slower when we have more input views, as more information is given and need to be fitted. Neural Body’s performance gets better, however their convergence speed is much slower than ours.

For generalizable methods, their performance improves significantly compared to single view input, as they can have more information to interpolate between views. However, their rendering quality is still much lower than ours, as shown in Fig. 3.

D.3. Ablation on the number of parts

To analyze the impact of the number of human parts, we conduct experiments on ”377” sequence to compare our method and another two variants of our method, one have fewer parts than ours and one have more parts. The detailed configuration for these two variants can be found in Table 9 and Table 10. They are both trained for the same amount of time and we keep their model size approximately the same. We find that these two variants both show lower performance, compared to ours. The quantitative results are shown in Table 11.

D.4. Ablation on the method used for aggregation

In addition, to perform the max pooling among predictions of human parts, we also tried other strategies for aggregation, including direct averaging, using the inverse of distance to this part to do a weighted sum, and using the closest part. We find that these aggregation strategies do not perform better than the max pooling, as shown in Table 12.

D.5. Ablation on the model size

We try to reduce our model’s size to a smaller size by decreasing the size of the hash table, and the resulting model has a model size of 13.4M. After training on the ”377” dataset for 5 minutes given monocular input, we find it still shows a PSNR/SSIM/LPIPS* = 31.13/0.98/31.91, which is still competitive compared to the baseline methods.

D.6. Ablation on the number of frames

The number of frames chosen for training also will affect the training effect. If the number of frames is too small, then the model cannot get enough information to reconstruct the human model. If we have too many frames for training, there would be more information to be fitted. We conduct experiments on the different choices of training frames by alternating the frame interval of the ”377” sequence. We choose the same frames as previous experiments for testing. We find that increasing or decreasing the frame count will affect the training greatly. The results can be found in Table 13.

		ZJU-MoCap					
			377			386	
Methods	Training time	PSNR ↑	SSIM ↑	LPIPS* ↓	PSNR ↑	SSIM ↑	LPIPS* ↓
Ours	∼5 minutes	31.36	0.979	26.03	33.53	0.977	33.02
HumanNeRF	∼10 hours	31.12	0.977	22.80	33.31	0.973	33.48
AS	∼10 hours	31.22	0.983	26.20	33.18	0.981	33.85
NB	∼10 hours	29.50	0.970	28.32	31.05	0.970	39.87
AN	∼10 hours	29.91	0.971	32.89	31.69	0.970	44.45
NHP	∼10 hours pre-training, ∼1 hour fine-tuning	27.67	0.957	60.08	30.62	0.965	55.81
PixelNeRF	∼10 hours pre-training, ∼1 hour fine-tuning	23.94	0.899	112.83	27.10	0.921	99.42
			387			392	
Methods	Training time	PSNR ↑	SSIM ↑	LPIPS* ↓	PSNR ↑	SSIM ↑	LPIPS* ↓
Ours	∼5 minutes	28.11	0.963	46.96	32.03	0.973	39.30
HumanNeRF	∼10 hours	28.27	0.962	38.89	31.34	0.971	33.57
AS	∼10 hours	27.43	0.968	43.84	30.96	0.976	40.31
NB	∼10 hours	27.14	0.955	47.63	28.38	0.965	44.35
AN	∼10 hours	27.03	0.957	54.01	31.18	0.968	47.65
NHP	∼10 hours pre-training, ∼1 hour fine-tuning	26.23	0.952	69.99	29.30	0.956	66.63
PixelNeRF	∼10 hours pre-training, ∼1 hour fine-tuning	24.37	0.889	121.36	24.69	0.886	126.65
			393			394	
Methods	Training time	PSNR ↑	SSIM ↑	LPIPS* ↓	PSNR ↑	SSIM ↑	LPIPS* ↓
Ours	∼5 minutes	29.55	0.964	46.29	31.46	0.969	39.10
HumanNeRF	∼10 hours	29.19	0.964	36.88	30.74	0.966	34.67
AS	∼10 hours	29.00	0.970	42.07	30.50	0.973	37.10
NB	∼10 hours	28.38	0.958	49.55	29.73	0.963	45.11
AN	∼10 hours	28.55	0.959	52.86	30.28	0.964	49.47
NHP	∼10 hours pre-training, ∼1 hour fine-tuning	27.13	0.949	70.47	28.53	0.951	65.65
PixelNeRF	∼10 hours pre-training, ∼1 hour fine-tuning	23.63	0.873	138.86	24.53	0.881	132.03

Table 6. Full quantitative comparison results on ZJU-MoCap.

		MonoCap					
			Lan			Marc	
Methods	Training time	PSNR ↑	SSIM ↑	LPIPS* ↓	PSNR ↑	SSIM ↑	LPIPS* ↓
Ours	∼5 minutes	32.78	0.987	17.13	33.84	0.989	16.92
HumanNeRF	∼10 hours	33.50	0.989	13.40	34.66	0.990	16.47
AS	∼10 hours	32.84	0.989	10.56	34.13	0.990	14.01
NB	∼10 hours	33.05	0.987	16.36	34.68	0.988	17.52
AN	∼10 hours	31.40	0.986	18.25	30.81	0.983	24.16
NHP	∼10 hours pre-training, ∼1 hour fine-tuning	29.42	0.975	31.74	30.56	0.972	39.98
PixelNeRF	∼10 hours pre-training, ∼1 hour fine-tuning	26.75	0.956	48.05	28.25	0.964	43.58
			Olek			Vlad	
Methods	Training time	PSNR ↑	SSIM ↑	LPIPS* ↓	PSNR ↑	SSIM ↑	LPIPS* ↓
Ours	∼5 minutes	34.95	0.990	13.93	28.88	0.984	18.72
HumanNeRF	∼10 hours	34.08	0.989	14.36	28.49	0.981	17.86
AS	∼10 hours	34.13	0.989	11.41	28.80	0.983	16.75
NB	∼10 hours	33.30	0.987	14.28	28.39	0.982	18.65
AN	∼10 hours	34.18	0.988	15.47	27.90	0.981	19.98
NHP	∼10 hours pre-training, ∼1 hour fine-tuning	33.78	0.988	16.12	28.26	0.983	20.70
PixelNeRF	∼10 hours pre-training, ∼1 hour fine-tuning	27.56	0.964	37.35	23.16	0.956	46.93

Table 7. Full quantitative comparison results on MonoCap.

References

[1] Jonathan T Barron, Ben Mildenhall, Dor Verbin, Pratul P Srinivasan, and Peter Hedman. Mip-nerf 360: Unbounded anti-aliased neural radiance fields. *arXiv preprint arXiv:2111.12077*, 2021. 2

[2] Marc Habermann, Lingjie Liu, Weipeng Xu, Michael Zoll-

	Training time	PSNR	SSIM	LPIPS*
Ours	∼13 minutes	32.55	0.981	26.5
HumanNeRF	∼10 hours	32.28	0.982	19.6
AS	∼10 hours	32.63	0.983	32.0
NB	∼10 hours	32.99	0.983	26.8
AN	∼10 hours	32.31	0.98	32.2
NHP	∼10 hours pre-training, ∼1 hour fine-tuning	31.73	0.976	41.1
PixelNeRF	∼10 hours pre-training, ∼1 hour fine-tuning	24.61	0.917	88.2

Table 8. Comparison of different methods trained on 4 views from 377 Dataset.

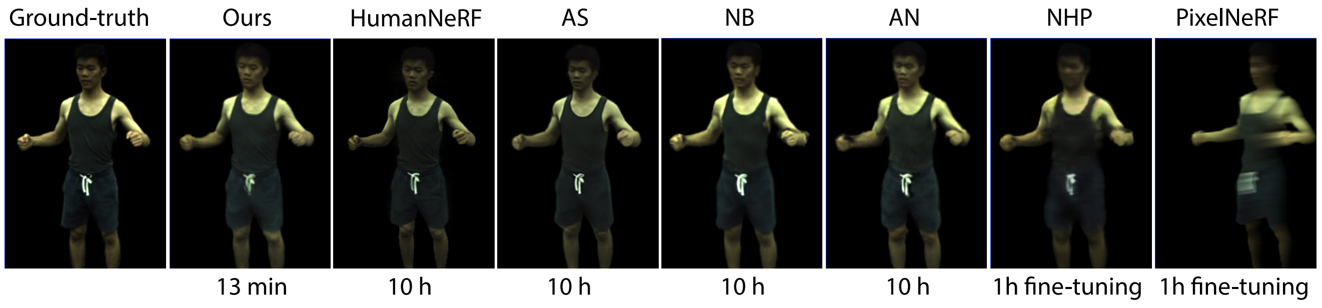


Figure 3. Qualitative results of methods trained with 4 views on the "377" sequence of the ZJU-MoCap dataset.

body	0, 3, 6, 9, 13, 14, 16, 17, 18, 19, 20, 21, 22, 23
leg	1, 2, 4, 5, 7, 8, 10, 11
head	12, 15

Table 9. Detailed conversion table for "part3" variant.

body	0, 3, 6, 9, 13, 14
left leg	1, 4, 7, 10
right leg	2, 5, 8, 11
head	12, 15
left arm	16, 18, 20, 22
right arm	17, 19, 21, 23

Table 10. Detailed conversion table for "part6" variant.

	PSNR	SSIM	LPIPS*	Num. of parameters
Ours	32.09	0.982	23.47	286M
Ours, 3 parts	31.78	0.980	26.13	278M
Ours, 6 parts	31.73	0.979	25.35	298M

Table 11. Performance of our method with different number of parts. All methods are trained for 5 minutes.

	PSNR	SSIM	LPIPS*
Ours, mean	28.65	0.974	39.33
Ours, dist weighted sum	27.33	0.958	63.97
Ours, closest part	26.9	0.956	64.31

Table 12. Comparison of different method to perform aggregation.

	PSNR	SSIM	LPIPS*
Ours, 250 frames	31.78	0.980	25.50
Ours, 100 frames	32.09	0.982	23.47
Ours, 50 frames	31.78	0.979	25.84

Table 13. Ablation studies on the number of frames. We train all models for 5 minutes.

hoefer, Gerard Pons-Moll, and Christian Theobalt. Real-time deep dynamic characters. In *SIGGRAPH*, 2021. 3

[3] Marc Habermann, Weipeng Xu, Michael Zollhofer, Gerard Pons-Moll, and Christian Theobalt. Deepcap: Monocular human performance capture using weak supervision. In *CVPR*, 2020. 3

[4] Youngjoong Kwon, Dahun Kim, Duygu Ceylan, and Henry Fuchs. Neural human performer: Learning generalizable radiance fields for human performance rendering. *Advances in Neural Information Processing Systems*, 34, 2021. 3, 4

[5] Matthew Loper, Naureen Mahmood, Javier Romero, Gerard Pons-Moll, and Michael J Black. Smpl: A skinned multi-person linear model. *ACM TOG*, 2015. 1, 3

[6] 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

[7] Thomas Müller, Alex Evans, Christoph Schied, and Alexander Keller. Instant neural graphics primitives with a multiresolution hash encoding. In *SIGGRAPH*, 2022. 2

[8] Michael Oechsle, Songyou Peng, and Andreas Geiger.

Unisurf: Unifying neural implicit surfaces and radiance fields for multi-view reconstruction. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 5589–5599, 2021. 1

[9] Sida Peng, Junting Dong, Qianqian Wang, Shangzhan Zhang, Qing Shuai, Xiaowei Zhou, and Hujun Bao. Animatable neural radiance fields for modeling dynamic human bodies. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 14314–14323, 2021. 1, 3

[10] Sida Peng, Shangzhan Zhang, Zhen Xu, Chen Geng, Boyi Jiang, Hujun Bao, and Xiaowei Zhou. Animatable neural implicit surfaces for creating avatars from videos. *arXiv preprint arXiv:2203.08133*, 2022. 3

[11] 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 *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9054–9063, 2021. 3

[12] Nikhila Ravi, Jeremy Reizenstein, David Novotny, Taylor Gordon, Wan-Yen Lo, Justin Johnson, and Georgia Gkioxari. Accelerating 3d deep learning with pytorch3d. *arXiv:2007.08501*, 2020. 3

[13] Stéfan van der Walt, Johannes L. Schönberger, Juan Nunez-Iglesias, François Boulogne, Joshua D. Warner, Neil Yager, Emmanuelle Gouillart, Tony Yu, and the scikit-image contributors. scikit-image: image processing in Python. *PeerJ*, 2:e453, 6 2014. 3

[14] Chung-Yi Weng, Brian Curless, Pratul P Srinivasan, Jonathan T Barron, and Ira Kemelmacher-Shlizerman. Humannerf: Free-viewpoint rendering of moving people from monocular video. *arXiv preprint arXiv:2201.04127*, 2022. 3

[15] Alex Yu, Vickie Ye, Matthew Tancik, and Angjoo Kanazawa. pixelnerf: Neural radiance fields from one or few images. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4578–4587, 2021. 3, 4

[16] Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *CVPR*, 2018. 3