Supplementary for ARAH: Animatable Volume Rendering of Articulated Human SDFs

Shaofei Wang[1], Katja Schwarz[2,3], Andreas Geiger[2,3], and Siyu Tang[1]

[1] ETH Zürich
[2] Max Planck Institute for Intelligent Systems, Tübingen
[3] University of Tübingen

Abstract. In this supplementary material, we first provide additional information on loss terms mentioned in the main paper. Secondly, we introduce the network architecture we used for forward LBS, canonical SDF, and canonical color networks. Next, we derive implicit gradients for forward LBS root-finding and our joint root-finding. We then present ablation studies of our approach. Further, we describe implementation details and how we set up baselines on the ZJU-MoCap dataset. Finally, we present additional results and discuss the limitations of our approach.

A Loss Definition

In Section 3.5 of the main paper, we define the loss terms as follows

$$\mathcal{L} = \lambda_C \cdot \mathcal{L}_C + \lambda_E \cdot \mathcal{L}_E + \lambda_O \cdot \mathcal{L}_O + \lambda_I \cdot \mathcal{L}_I + \lambda_S \cdot \mathcal{L}_S \tag{1}$$

In this section, we elaborate on how each loss term is defined. Let $I_p \in [0,1]^3$ denote the ground-truth RGB value of a pixel p. Further, let P denote the set of all pixels sampled from an image.

RGB Color Loss: The RGB color loss is defined as

$$\mathcal{L}_C = \frac{1}{|P|} \sum_{p \in P} \left| f_{\sigma_c}(\hat{\mathbf{x}}^{(p)}, \mathbf{n}^{(p)}, \mathbf{v}^{(p)}, \mathbf{z}, \mathcal{Z}) - I_p \right| \tag{2}$$

Eikonal Regularization: We sample 1024 points, denoted as $\hat{\mathbf{X}}_{\text{eik}}$, in the range $[-1,1]^3$ in canonical space, and compute Eikonal loss [4] as follows:

$$\mathcal{L}_E = \frac{1}{|P|} \sum_{\hat{\mathbf{x}} \in \hat{\mathbf{X}}_{\text{eik}}} \left| \|\nabla_{\hat{\mathbf{x}}} f_{\sigma_f}(\hat{\mathbf{x}})\|_2 - 1 \right| \tag{3}$$

Off-surface Point Loss: In canonical space, we sample 1024 points whose distance to the canonical SMPL mesh is greater than 20cm. Let $\hat{\mathbf{X}}_{\text{off}}$ denote these sampled points, we compute the off-surface point loss as

$$\mathcal{L}_O = \frac{1}{|P|} \sum_{\hat{\mathbf{x}} \in \hat{\mathbf{X}}_{\text{off}}} \exp\left(-1e^2 \cdot f_{\sigma_f}(\hat{\mathbf{x}})\right) \tag{4}$$

Inside Point Loss: In canonical space, we sample 1024 points that are inside the canonical SMPL mesh and whose distance to the SMPL surface is greater than 1cm. Let $\hat{\mathbf{X}}_{\text{in}}$ denote these sampled points, we compute the inside point loss as

$$\mathcal{L}_I = \frac{1}{|P|} \sum_{\hat{\mathbf{x}} \in \hat{\mathbf{X}}_{\text{in}}} \text{sigmoid}\left(5e^3 \cdot f_{\sigma_f}(\hat{\mathbf{x}})\right) \tag{5}$$

Skinning Loss: Finally, in canonical space, we sample 1024 points on the canonical SMPL surface, $\hat{\mathbf{X}}_{\text{S}}$, and regularize the forward LBS network with the corresponding SMPL skinning weights $\mathbf{W} = \{\mathbf{w}\}$:

$$\mathcal{L}_S = \frac{1}{|P|} \sum_{\substack{\hat{\mathbf{x}} \in \hat{\mathbf{X}}_{\text{S}} \\ \mathbf{w} \in \mathbf{W}}} \sum_{i=1}^{i=24} \left| f_{\sigma_\omega}(\hat{\mathbf{x}})_i - \mathbf{w}_i \right| \tag{6}$$

We set $\lambda_C = 3e^1, \lambda_E = 5e^1, \lambda_O = 1e^2, \lambda_I = \lambda_S = 10$ throughout all experiments.
Mask Loss: As described in Section 3.5 of the main paper, our volume rendering formulation does not need explicit mask loss. Here we describe the mask loss from [23] which we use in the ablation study on surface rendering (Section F). Given the camera ray $\mathbf{r}^{(p)} = (\mathbf{c}, \mathbf{v}^{(p)})$ of a specific pixel p, we first define $S(\alpha, \mathbf{c}, \mathbf{v}^{(p)}) = \text{sigmoid}(-\alpha \min_{d \geq 0} f_{\sigma_f}(LBS_{\sigma_\omega}^{-1}(\mathbf{c} + d\mathbf{v}^{(p)}))$, *i.e.* the Sigmoid of the minimal SDF along a ray. In practice we sample 100 ds uniformly between $[d_{\min}, d_{\max}]$ along the ray, where $d_{\min}$ and $d_{\max}$ are determined by the bounding box of the registered SMPL mesh. α is a learnable scalar parameter.

Let $O_p \in \{0, 1\}$ denote the foreground mask value (0 indicates background and 1 indicates foreground) of a pixel p. Further, let P_{in} denote the set of pixels for which ray-intersection with the iso-surface of neural SDF is found and $O_p = 1$, while $P_{out} = P \setminus P_{in}$ is the set of pixels for which no ray-intersection with the iso-surface of neural SDF is found or $O_p = 0$. The mask loss is defined as

$$\mathcal{L}_M = \frac{1}{\alpha |P|} \sum_{p \in P_{out}} \text{BCE}(O_p, S(\alpha, \mathbf{c}, \mathbf{v}^{(p)})) \tag{7}$$

where $\text{BCE}(\cdot)$ denotes binary cross entropy loss. We set the weight of $\mathcal{L}_M$ to be $3e^3$ and add this loss term to Eq. (1) for our surface rendering baseline in Section F.

B Network Architectures

In this section, we describe detailed network architectures for the forward LBS network f_{σ_ω}, the SDF network f_{σ_f} and the color network f_{σ_c} introduced in Sections 3.1-3.2 of the main paper.

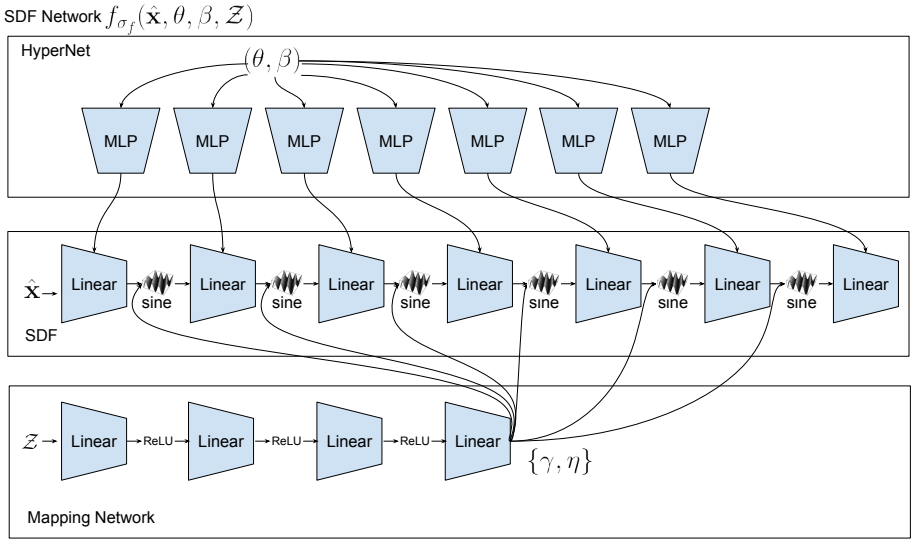

Fig. 1: **Network Architecture for the SDF Network.** Our SDF network builds upon MetaAvatar [19] which uses a hypernetwork (top) that conditions on local body poses and shape (θ, β), and predicts the parameters of a neural SDF with periodic activation (middle). Since MetaAvatar does not model per-frame latent codes, we add a mapping network (bottom) that maps the per-frame latent code $\mathcal{Z}$ to scaling factors $\{\gamma\}$ and offsets $\{\eta\}$ which are used to modulate the outputs from each linear layer of the neural SDF, except for the last layer.

B.1 Forward LBS Network

We use the same forward LBS network as [3], which consists of 4 hidden layers with 128 channels and weight normalization [15]. It uses Softplus activation with $\beta = 100$. f_{σ_ω} only takes query points in canonical space as inputs and does not have any conditional inputs.

To initialize this forward LBS network, we meta learn the network on skinning weights of canonical meshes from the CAPE [8] dataset. Specifically, we use Reptile [10] with 24 inner steps. The inner learning rate is set to $1e^{-4}$ while the outer learning rate is set to $1e^{-5}$. Adam [6] optimizer is used for both the inner and the outer loop. We train with a batch size of 4 for 100k steps of the outer loop. We use the resulting model as the initialization for our per-subject optimization on the ZJU-MoCap [13] dataset.

B.2 Canonical SDF Network

We describe our canonical SDF network in Fig. 1. The hypernetwork (top) and neural SDF (middle) are initialized with MetaAvatar [19] pre-trained on the CAPE dataset. Note that the SDF network from MetaAvatar can be trained

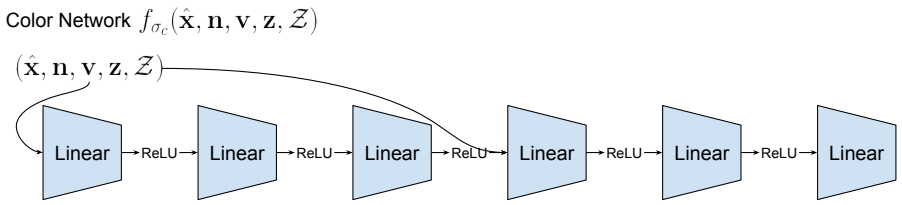

Fig. 2: **Network Architecture for the Color Network.** The color network takes canonicalized query points $\hat{\mathbf{x}}$, normal vectors $\mathbf{n}$, viewing directions $\mathbf{v}$, an SDF feature $\mathbf{z}$, and a per-frame latent code $\mathcal{Z}$ as inputs.

with canonical meshes only and does not need any posed meshes as supervision. Each MLP of the hypernetwork (top) consists of one hidden layer with 256 channels and uses ReLU activation. The neural SDF (middle) consists of 5 hidden layers with 256 channels and uses a periodic activation [16]. In addition to the MetaAvatar SDF, we add a mapping network [2, 14] which consists of 2 hidden layers with 256 channels and a ReLU activation. It maps the per-frame latent code $\mathcal{Z}$ to scaling factors and offsets that modulate the outputs from each layer of the neural SDF. We initialize the last layer of the mapping network to predict scaling factors with value 1 and offsets with value 0.

B.3 Canonical Color Network

We describe our canonical color network in Fig. 2. The network consists of 4 hidden layers with 256 channels and ReLU activation. The inputs to the network are also concatenated with activations of the third layer and fed into the fourth layer together.

C Implicit Gradients

In this section, we describe how to compute gradients of the root-finding solutions wrt. the forward LBS network and the SDF network. In the main paper, we use our novel joint root-finding algorithm to find the surface point and sample points around the surface point; these sampled points, along with the surface point, are mapped to canonical space via iterative root-finding [3]. Section C.1 describes how to differentiate through these points to compute gradients wrt. the forward LBS network. Section C.2 describes how to compute gradients wrt. the forward LBS network and the SDF network given the surface point and its correspondence. Section C.1 is used for volume rendering, which is described in Section 3.4 of the main paper. Section C.2 is used for surface rendering, which is one of our ablation baselines in Section F.

C.1 Implicit Gradients for Forward LBS

Here we follow [3] and describe how to compute implicit gradients for the forward LBS network given samples on camera rays and their canonical correspondences. Denoting sampled points in observation space as $\bar{\mathbf{X}} = \{\bar{\mathbf{x}}\}_{i=1}^{N}$, and their canonical correspondences obtained by iterative root-finding [3] as $\hat{\mathbf{X}}^* = \{\hat{\mathbf{x}}^*\}_{i=1}^{N}$, they should satisfy the following condition

$$LBS_{\sigma_\omega}(\hat{\mathbf{x}}^{*(i)}) - \bar{\mathbf{x}}^{(i)} = 0, \quad \forall i = 1, \ldots, N \tag{8}$$

As done in [23], by applying implicit differentiation, we obtain a differentiable point sample $\hat{\mathbf{x}}$ as

$$\hat{\mathbf{x}} = \hat{\mathbf{x}}^* - (\mathbf{J}^*)^{-1} \cdot \left(LBS_{\sigma_\omega}(\hat{\mathbf{x}}^{*(i)}) - \bar{\mathbf{x}}^{(i)} \right) \tag{9}$$

where $\mathbf{J}^* = \frac{\partial LBS_{\sigma_\omega}}{\partial \hat{\mathbf{x}}}(\hat{\mathbf{x}}^*)$. $\hat{\mathbf{x}}^*$ and $\mathbf{J}^*$ are detached from the computational graph such that no gradient will flow through them. These differentiable samples can be used as inputs to the SDF and color networks. Gradients wrt. σ_ω are computed from photometric loss Eq. (2) via standard back-propagation. Taking the derivative wrt. σ_ω for both sides of Eq. (9) results in the same analytical gradient defined in Eq. (14) of [3].

Pose and Shape Optimization: We note that implicit gradients can also be back-propagated to SMPL parameters $\{\theta, \beta\}$ as the SMPL model is fully differentiable. We found pose and shape optimization particularly helpful when SMPL estimations are noisy, *e.g.* those estimated from monocular videos. In Fig. 3 we show a qualitative sample on the People Snapshot dataset [1] where the pose is improved while the resulting model also achieves better visual quality.

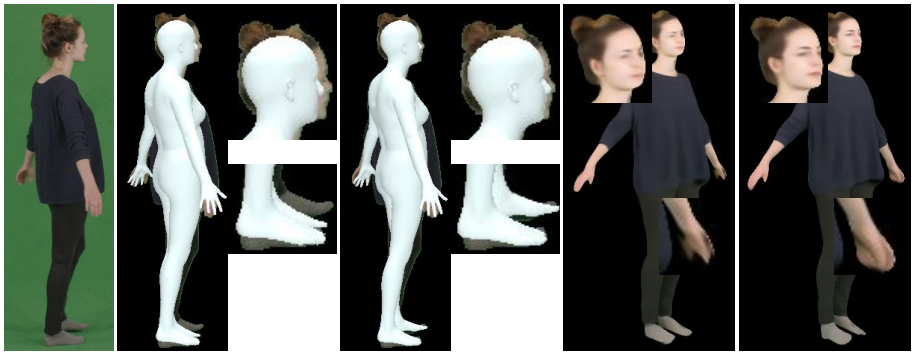

Fig. 3: **Result of Pose and Shape Optimization.** We can improve the noisy SMPL estimations on training poses with implicit gradients and improve the rendering quality on unseen poses (see Unseen w. opt.).

C.2 Implicit Gradients for Joint Root-finding

Now we derive implicit gradients for our joint root-finding algorithm. We denote the joint vector-valued function of the ray-surface intersection and forward LBS as $g_{\sigma_f,\sigma_\omega}(\hat{\mathbf{x}}, d)$. The joint root-finding problem is

$$g_{\sigma_f,\sigma_\omega}(\hat{\mathbf{x}}, d) = \begin{bmatrix} f_{\sigma_f}(\hat{\mathbf{x}}) \\ LBS_{\sigma_\omega}(\hat{\mathbf{x}}) - (\mathbf{c} + \mathbf{v} \cdot d) \end{bmatrix} = \mathbf{0} \tag{10}$$

with a slight abuse of notation, we denote the iso-surface point as $\hat{\mathbf{x}}^*$ and their corresponding depth in observation space as d^*. We follow [23] and use implicit differentiation to obtain a differentiable point sample $\hat{\mathbf{x}}$ and a depth sample d:

$$\begin{bmatrix} \hat{\mathbf{x}} \\ d \end{bmatrix} = \begin{bmatrix} \hat{\mathbf{x}}^* \\ d^* \end{bmatrix} - (\mathbf{J}^*)^{-1} \cdot g_{\sigma_f,\sigma_\omega}(\hat{\mathbf{x}}^*, d^*) \tag{11}$$

where $\mathbf{J}^*$ is defined as

$$\mathbf{J}^* = \begin{bmatrix} \frac{\partial f_{\sigma_f}}{\partial \hat{\mathbf{x}}}(\hat{\mathbf{x}}^*) & 0 \\ \frac{\partial LBS_{\sigma_\omega}}{\partial \hat{\mathbf{x}}}(\hat{\mathbf{x}}^*) & -\mathbf{v} \end{bmatrix} \tag{12}$$

Similar to Section C.1, these differentiable samples can be used as inputs to the SDF and color networks and gradients wrt. σ_f, σ_ω can be computed from the photometric loss Eq. (2).

D Implementation Details

We use Adam [6] to optimize our models and the per-frame latent codes $\{\mathcal{Z}\}$. We initialize the SDF network with MetaAvatar [19] and set the learning rate to $1e^{-6}$ as suggested in [19]. For the remaining models and the latent codes, we use a learning rate of $1e^{-4}$. We apply weight decay with a weight of 0.05 to the per-frame latent codes.

We train our models with a batch size of 4 and 2048 rays per batch, with 1024 rays sampled from the foreground mask and 1024 rays sampled from the background. As mentioned in Section 3.4 of the main paper, we sample 16 near and 16 far surface points for rays that intersect with a surface and 64 points for rays that do not intersect with a surface. Our model is trained for 250 epochs (except for sequence 313 which we trained for 1250 epochs, due to its training frames being much fewer than other sequences), which corresponds to 60k-80k iterations depending on the amount of training data. This takes about 1.5 days on 4 NVIDIA 2080 Ti GPUs. During training, we follow [20] and add normally distributed noise with zero mean and a standard deviation of 0.1 to the input θ of the SDF network. This noise ensures that the canonical SDF does not fail when given extreme out-of-distribution poses. We also augment the input viewing directions to the color network during training. We do so by randomly applying

roll/pitch/yaw rotations sampled from a normal distribution with zero mean and a standard deviation of 45° to the viewing direction, but reject augmentation in which the angle between the estimated surface normal and the negated augmented viewing direction is greater than 90 degrees.

For inference, we follow [12,13] and crop an enlarged bounding box around the projected SMPL mesh on the image plane and render only pixels inside the bounding box. For unseen test poses we follow the practice of [12, 13] and use the latent code $\mathcal{Z}$ of the last training frame as the input. The rendering time of a 512×512 image is about 10-20 seconds, depending on the bounding box size of the person. In this process, the proposed joint root-finding algorithm takes about 1 second.

E Implementation Details for Baselines

In this section, we describe the implementation details of the baselines from the main paper, *i.e.* Neural Body [13], Ani-NeRF [12], and A-NeRF [17].

E.1 Neural Body

For quantitative evaluation, we use the official results provided by the Neural Body website. For generating rendering results and geometries, we use the official code of Neural Body and their pre-trained models without modification.

E.2 Animatable NeRF (Ani-NeRF)

For quantitative evaluation, we use the official code and pre-trained models when possible, *i.e.* for sequences 313, 315, 377, and 386. For the remaining sequences that the official code does not provide pre-trained models, we train models using the default hyperparameters that were applied to sequences 313, 315, 377, and 386.

We note that when reconstructing geometry on the training poses, Neural Body and Ani-NeRF compute visual hulls from ground-truth masks of training views and set density values outside the visual hulls to 0. This removes extraneous geometry blobs from reconstructions by Neural Body and Ani-NeRF. When testing on unseen poses, we disable the mask usage, as, by definition of the task, we do not have any image as input.

E.3 A-NeRF

For A-NeRF, we follow the author's suggestions to 1) use a bigger foreground mask for ray sampling, 2) enable background estimation in the official code, and 3) use L2 loss instead of L1 loss. The learned models give reasonable novel view synthesis results on training poses (Fig. 6) but cannot generalize to unseen poses (Fig. 7). We hypothesize that this is because training poses on the ZJU-MoCap dataset are extremely limited, and A-NeRF uses only keypoints instead of surface

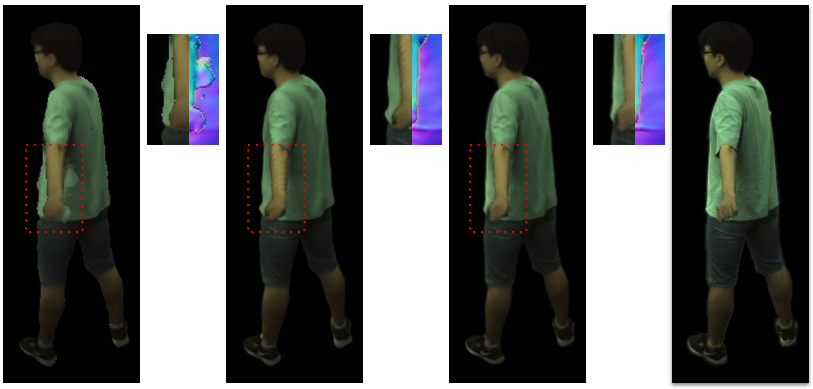

Fig. 4: **Ablation on ray sampling strategies**. We observe severe geometric artifacts with models trained with surface rendering. A simple uniform sampling strategy (as used in [12, 13]) produces stratified artifacts due to the discretized sampling. In contrast, our proposed approach does not suffer from these problems and achieves better result.

models to construct their conditional inputs to NeRF networks. The lack of a surface model makes it easy for A-NeRF to confuse background and foreground, resulting in obvious floating blob artifacts. These artifacts are amplified when training poses are limited, making the generalization result of A-NeRF on the ZJU-MoCap dataset the worst among the baselines.

F Ablation Study

In this section, we ablate on ray sampling strategies as well as canonicalization strategies. We conduct an ablation on sequence 313. Metrics on all novel views of training poses are reported.

F.1 Ablation on Ray Sampling Strategies

We compare our proposed ray sampling strategy to surface rendering and uniform sampling with 64 samples on the novel view synthesis task (Fig 4). As discussed in the main paper, we did not use more sophisticated hierarchical sampling strategies [9, 18, 22] due to the computational cost of running the iterative root-finding [3] on dense samples and the memory cost for running additional forward/backward passes through the LBS network.

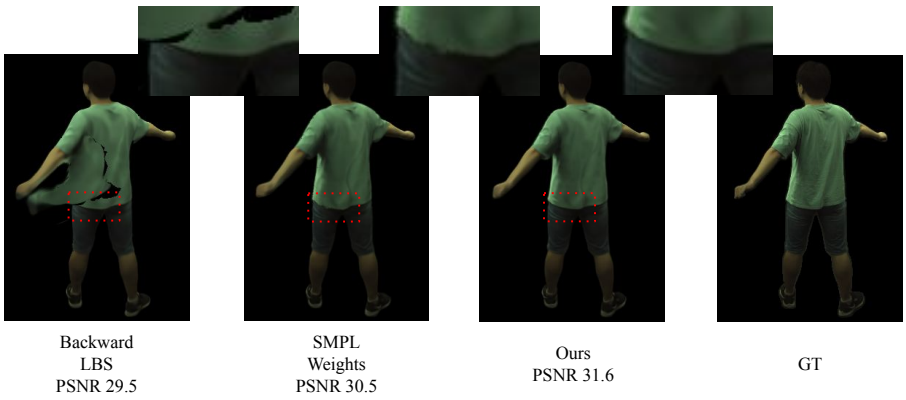

Fig. 5: **Ablation on Learned LBS networks**. Backward LBS has difficulties with learning skinning weights for points far from the surface, resulting in artifacts under specific poses. Canonicalization with deterministic SMPL weights results in discretized artifacts on the cloth surface. In contrast, our approach does not suffer from these problems.

F.2 Ablation on Learned forward LBS

In this subsection, we replace our learned forward LBS with (1) a backward LBS network that conditions on local body poses θ, and (2) a deterministic LBS with nearest neighbor SMPL skinning weights. For the learned backward LBS, we always canonicalize the query points using the SMPL global translation and rotation before querying the LBS network. We also sample points on the transformed SMPL meshes and supervise the backward LBS network with corresponding skinning weights using Eq. (6). We show qualitative results in Fig. 5.

F.3 Ablation on Root-finding Initialization

To ablate the effect of multiple initializations for root-finding, we add additional initializations from the nearest 2 SMPL bones but do not observe any noticeable change in metrics. We report PSNR/SSIM/LPIPS as: single initialization - 31.6/0.973/0.050, 2 more initializations: 31.5/0.972/0.049. Also, adding more initializations for root-finding drastically increases memory/time consumption, we thus decide to use only a single initialization for root-finding in our approach.

G Additional Quantitative Results

We present complete evaluation metrics including PSNR, SSIM, LPIPS on the test poses of the ZJU-MoCap [13] dataset in Table 1.

We also report quantitative results on the H36M dataset [5], following the testing protocols proposed by [12] in Table 2.

Table 1: **Complete evaluation results on novel pose synthesis.** PSNR, SSIM, LPIPS are reported for the test poses of the ZJU-MoCap dataset.

	313			315			377		
Method	PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓
NB	24.1	0.908	0.126	19.8	0.867	0.152	24.2	0.917	0.119
Ani-N	23.9	0.909	0.115	19.2	0.855	0.167	22.6	0.900	0.153
A-NeRF	22.0	0.855	0.209	18.7	0.810	0.232	22.6	0.890	0.165
Ours	**24.4**	**0.914**	**0.092**	**20.0**	**0.881**	**0.105**	**25.5**	**0.933**	**0.093**
	386			387			390		
Method	PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓
NB	26.1	0.894	0.171	22.7	0.902	0.135	24.2	0.882	0.164
Ani-N	25.5	0.884	0.187	23.1	0.906	0.145	23.9	0.887	0.173
A-NeRF	24.8	0.858	0.241	22.4	0.885	0.162	22.6	0.846	0.226
Ours	**27.0**	**0.910**	**0.127**	**24.2**	**0.917**	**0.099**	**24.8**	**0.896**	**0.126**
	392			393			394		
Method	PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓
NB	26.0	0.916	0.135	23.5	0.900	0.132	24.1	0.888	0.150
Ani-N	24.3	0.900	0.169	23.8	0.897	0.155	24.1	0.887	0.171
A-NeRF	23.7	0.886	0.183	22.1	0.875	0.175	22.7	0.861	0.199
Ours	**26.2**	**0.927**	**0.106**	**24.4**	**0.915**	**0.104**	**25.2**	**0.908**	**0.111**

Table 2: **Evaluation results on the H36M dataset.** Numbers of NARF [11] and Ani-N [12] are reported in [21].

	Training Poses						**Unseen Poses**					
	PSNR ↑			SSIM ↑			PSNR ↑			SSIM ↑		
	NARF	Ani-N	Ours	NARF	Ani-N	Ours	NARF	Ani-N	Ours	NARF	Ani-N	Ours
S1	21.41	22.05	**24.45**	0.891	0.888	**0.919**	20.19	21.37	**23.08**	0.864	0.868	**0.899**
S5	**25.24**	23.27	24.54	0.914	0.892	**0.918**	**23.91**	22.29	22.79	**0.891**	0.875	0.890
S6	21.47	21.13	**24.61**	0.871	0.854	**0.903**	22.47	22.59	**24.04**	0.883	0.884	**0.900**
S7	21.36	22.50	**24.31**	0.899	0.890	**0.919**	20.66	22.22	**22.58**	0.876	0.878	**0.891**
S8	22.03	22.75	**24.02**	0.904	0.898	**0.921**	21.09	21.78	**22.34**	0.887	0.882	**0.896**
S9	25.11	24.72	**26.20**	0.906	0.908	**0.924**	23.61	23.72	**24.36**	0.881	0.886	**0.894**
S11	24.35	24.55	**25.43**	0.902	0.902	**0.921**	23.95	23.91	**24.78**	0.885	0.889	**0.902**
Average	23.00	23.00	**24.79**	0.898	0.890	**0.918**	22.27	22.55	**23.42**	0.881	0.880	**0.896**

H Additional Qualitative Results

H.1 Qualitative Results on ZJU-MoCap Training Poses

We present additional qualitative results on ZJU-MoCap training poses in Fig. 6. Due to better geometry constraints, our approach better captures cloth wrinkles, textures, and face details. We also avoid extraneous color blobs under novel views which all baselines suffer from.

H.2 Additional Qualitative Results on ZJU-MoCap Test Poses

We show additional qualitative results on ZJU-MoCap test poses in Fig. 7. Similar to the results presented in the main paper, A-NeRF and Neural Body do not generalize to these within-distribution poses. Ani-NeRF produces noisy rendering due to its inaccurate backward LBS network. Note that since these results are pose extrapolations, it is not possible to reproduce the exact color and texture of ground-truth images. Still, our approach does not suffer from the artifacts that baselines have demonstrated, resulting in better metrics, especially for LPIPS (Table 1). We present more qualitative results in the supplementary video.

H.3 Additional Qualitative Results on Out-of-distribution Poses

We show additional qualitative results on out-of-distribution poses [7] in Fig. 8. We present more results in the supplementary video.

H.4 Closest Training Poses to Out-of-distribution Poses

To further demonstrate the generalization ability of our approach, we also visualize the closest training pose from the ZJU-MoCap dataset to out-of-distribution test poses from the AIST++ dataset and the AMASS dataset in Fig. 9. To find the closest training pose to a test pose, we convert local poses (*i.e.* all pose vectors excluding global orientation) to a matrix representation and find the closest training pose with nearest neighbor search using the converted matrix representation.

H.5 Qualitative Results on Models Trained with Monocular Videos

In this subsection, we present models trained on monocular videos. For this monocular setup, we use only the first camera of the ZJU-MoCap dataset to train our models. We do not modify our approach and all hyperparameters remain the same as the multi-view setup. We train each model for 500 epochs on 500 frames of selected sequences in which the subjects do repetitive motions while rotating roughly 360 degrees. We animate the trained model with out-of-distribution poses from AIST++ [7]. Qualitative results are shown in Fig. 10. Even under this extreme setup, our approach can still learn avatars with plausible geometry/appearance and the avatars still generalize to out-of-distribution poses. For the complete animation sequences, please see our supplementary video.

I Limitations

As reported in Section D, our approach is relatively slow at inference time. The major bottlenecks are the iterative root-finding [3] and the volume rendering.

Another limitation is that neural rendering-based reconstruction methods tend to overfit the geometry to the texture, resulting in a reconstruction bias. As shown in Fig. 11, while NeRF-based baselines are unable to recover detailed wrinkles, SDF-based rendering (ours and NeuS) wrongfully reconstructs the stripes on the shirt as part of the geometry. Note that A-NeRF and Ani-NeRF also suffer from this kind of bias. Neural Body demonstrates less overfitting effects. We hypothesize that this is because the structured latent codes in Neural Body are local in space and thus give the color network more flexibility, making the density network less prone to overfitting. Still, Neural Body gives noisy reconstructions and cannot generalize to unseen poses. Resolving this reconstruction bias while maintaining a clean geometry is an interesting avenue for future research.

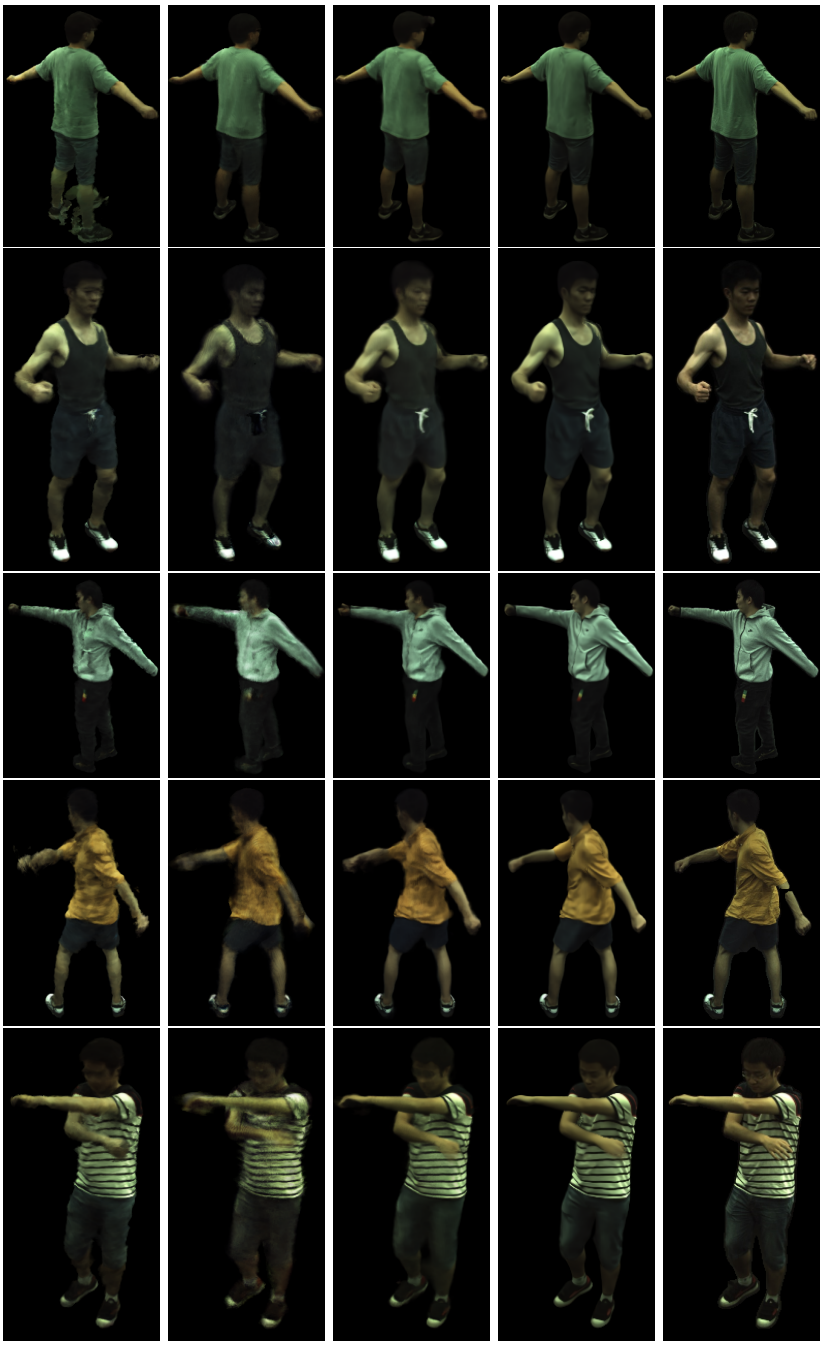

Fig. 6: **Novel View Synthesis Results** on the training poses of ZJU-MoCap.

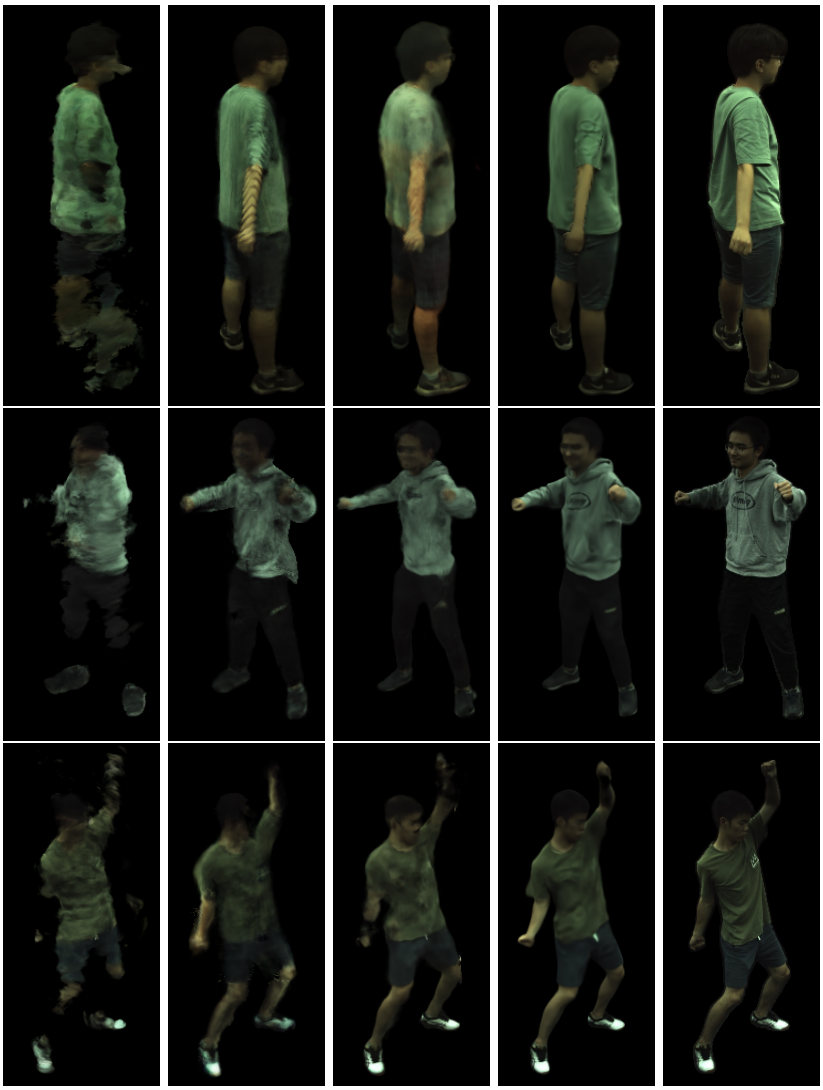

Fig. 7: **Additional Generalization Results on ZJU-MoCap Test Poses.**

Fig. 8: **Additional Generalization Results on Out-of-distribution Poses.** From top to bottom: Neural Body, Ani-NeRF, ours, and our geometry.

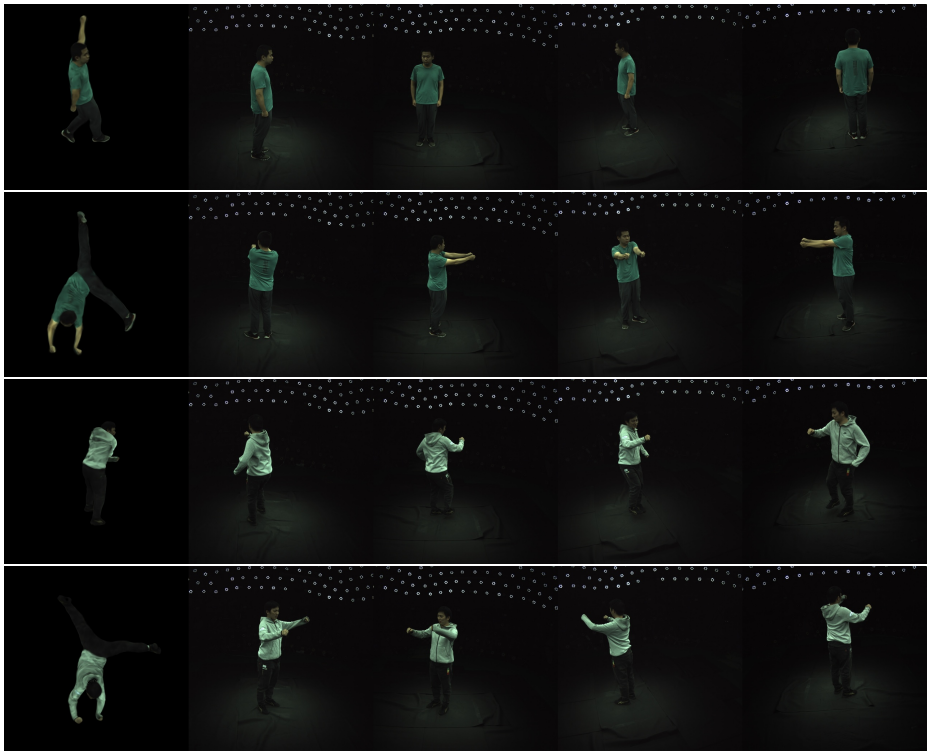

Fig. 9: **Closest Training Poses to Out-of-distribution Test Poses.** We show rendering results of out-of-distribution poses on the left-most column, while demonstrating 4 training images of the closest training pose to each of the test poses.

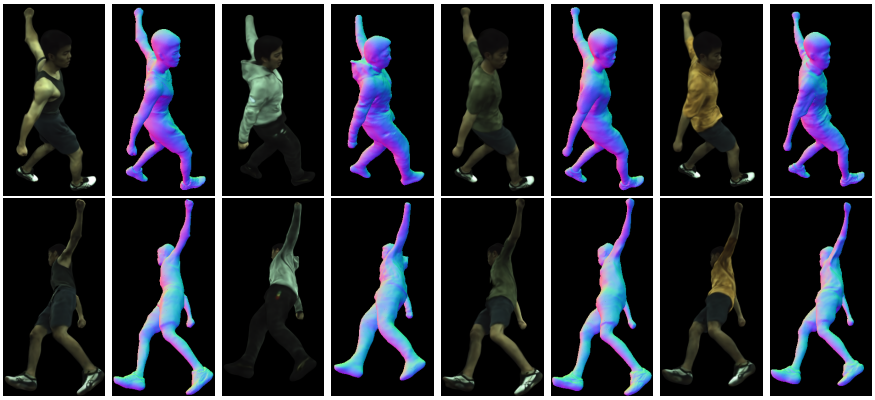

Fig. 10: **Generalization to AIST++ [7] Poses with Models Trained from Monocular Videos.**

Fig. 11: **Shape-Appearance Ambiguity**. The Neural Rendering-based reconstruction tends to bake complex textures into the geometry, resulting in a biased geometry reconstruction.

References

1. Alldieck, T., Magnor, M., Xu, W., Theobalt, C., Pons-Moll, G.: Video based reconstruction of 3d people models. In: Proc. of CVPR (2018) 5
2. Chan, E., Monteiro, M., Kellnhofer, P., Wu, J., Wetzstein, G.: pi-gan: Periodic implicit generative adversarial networks for 3d-aware image synthesis. In: Proc. of CVPR (2021) 4
3. Chen, X., Zheng, Y., Black, M., Hilliges, O., Geiger, A.: Snarf: Differentiable forward skinning for animating non-rigid neural implicit shapes. In: Proc. of ICCV (2021) 3, 4, 5, 8, 12
4. Gropp, A., Yariv, L., Haim, N., Atzmon, M., Lipman, Y.: Implicit geometric regularization for learning shapes. In: Proc. of ICML (2020) 1
5. Ionescu, C., Papava, D., Olaru, V., Sminchisescu, C.: Human3.6m: Large scale datasets and predictive methods for 3d human sensing in natural environments. IEEE Transactions on Pattern Analysis and Machine Intelligence **36**(7), 1325–1339 (jul 2014) 9
6. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. In: Proc. of ICLR (2015) 3, 6
7. Li, R., Yang, S., Ross, D.A., Kanazawa, A.: Ai choreographer: Music conditioned 3d dance generation with aist++. In: Proc. of ICCV (2021) 11, 16
8. Ma, Q., Yang, J., Ranjan, A., Pujades, S., Pons-Moll, G., Tang, S., Black, M.J.: Learning to dress 3D people in generative clothing. In: Proc. of CVPR (2020) 3
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: Proc. of ECCV (2020) 8
10. Nichol, A., Achiam, J., Schulman, J.: On first-order meta-learning algorithms. arXiv preprint arXiv:1803.02999 (2018) 3
11. Noguchi, A., Sun, X., Lin, S., Harada, T.: Neural articulated radiance field. In: Proc. of ICCV (2021) 10
12. Peng, S., Dong, J., Wang, Q., Zhang, S., Shuai, Q., Zhou, X., Bao, H.: Animatable neural radiance fields for modeling dynamic human bodies. In: Proc. of ICCV (2021) 7, 8, 9, 10
13. Peng, S., Zhang, Y., Xu, Y., Wang, Q., Shuai, Q., Bao, H., Zhou, X.: Neural body: Implicit neural representations with structured latent codes for novel view synthesis of dynamic humans. In: Proc. of CVPR (2021) 3, 7, 8, 9
14. Perez, E., Strub, F., de Vries, H., Dumoulin, V., Courville, A.C.: Film: Visual reasoning with a general conditioning layer. In: Proc. of AAAI (2018) 4
15. Salimans, T., Kingma, D.P.: Weight normalization: A simple reparameterization to accelerate training of deep neural networks. In: Proc. of NeurIPS (2016) 3
16. Sitzmann, V., Martel, J.N., Bergman, A.W., Lindell, D.B., Wetzstein, G.: Implicit neural representations with periodic activation functions. In: Proc. of NeurIPS (2020) 4
17. Su, S.Y., Yu, F., Zollhoefer, M., Rhodin, H.: A-neRF: Articulated neural radiance fields for learning human shape, appearance, and pose. In: Proc. of NeurIPS (2021) 7
18. Wang, P., Liu, L., Liu, Y., Theobalt, C., Komura, T., Wang, W.: Neus: Learning neural implicit surfaces by volume rendering for multi-view reconstruction. In: Proc. of NeurIPS (2021) 8
19. Wang, S., Mihajlovic, M., Ma, Q., Geiger, A., Tang, S.: Metaavatar: Learning animatable clothed human models from few depth images. In: Proc. of NeurIPS (2021) 3, 6

20. Xu, H., Alldieck, T., Sminchisescu, C.: H-neRF: Neural radiance fields for rendering and temporal reconstruction of humans in motion. In: Proc. of NeurIPS (2021) 6
21. Xu, T., Fujita, Y., Matsumoto, E.: Surface-aligned neural radiance fields for controllable 3d human synthesis. In: CVPR (2022) 10
22. Yariv, L., Gu, J., Kasten, Y., Lipman, Y.: Volume rendering of neural implicit surfaces. In: Proc. of NeurIPS (2021) 8
23. Yariv, L., Kasten, Y., Moran, D., Galun, M., Atzmon, M., Ronen, B., Lipman, Y.: Multiview neural surface reconstruction by disentangling geometry and appearance. In: Proc. of NeurIPS (2020) 2, 5, 6