Supplemental Material for
Structured Local Radiance Fields for Human Avatar Modeling

A. Overview

This supplementary document provides more discussions and experimental details. In Sec. B, we discuss in detail the differences between our method and state-of-the-art approaches. Details about network architecture are presented in Sec. C. In Sec. D we present more details about how we collect the data and how we conduct the experiments. We conduct additional experiments in Sec. E to further evaluate our method design. Finally, we discuss the limitations and potential future work in Sec. F. Please refer to the supplementary video for more visualizations.

B. More Discussion

Our method aims at creating a controllable 3D human character from RGB videos without pre-scanning a subject-specitic template. To better motivate our method and differentiate from existing approaches, we list the most related works below and discuss their limitations as well as our solution in this section.

Neural Body [9] attaches learnable latent codes to the vertices of SMPL model, and employs sparse 3D convolutions to diffuse the latent codes into a radiance field in the 3D space. This scheme shows impressive performance on novel view synthesis for human performance. However, it struggles with new pose syntheses, as shown in [8]. The main reason for this limitation is that 3D convolution is not equivalent to spatial changes of the structured latent code. In our method, we avoid the need for 3D convolutions and construct the radiance field by combining a set of localized ones, thus easily enable avatar animation by design.

Animatable NeRF [8] factorizes a deforming human body into a canonical radiance field and per-frame deformation fields that establish correspondences between the observations and the canonical space. The deformation field is generated through diffusing the input skeleton motion into the 3D space based on the learnable blending weights. Thanks to the explicit disentanglement of shape and motion, Animatable NeRF [8] is able to synthesize images for unseen poses. However, the motion representation is too simple to model the complex non-rigid deformations of clothes, which results into unrealistic, static texture and even severe artifacts when applying this method on loose clothes. In contrast, our method explicitly takes into account the non-rigid cloth deformation via coarse-to-fine decomposition, and demonstrates plausible animation results for human characters wearing dresses.

Neural Actor [5] shares a similar scheme with Animatable NeRF [8]: it also learns a neural radiance field in a canonical body pose, and use LBS to warp the canonical radiance field to represent the moving subject. Its main innovations are two fold: 1) Neural Actor learns pose-dependent non-rigid deformation that cannot be captured by standard skinning using a residual function, and 2) Neural Actor encodes appearance features on the 2D texture maps of the SMPL model to better capture dynamic details. Although this scheme shows impressive results in modeling the pose-dependent appearance details like the cloth wrinkles, it only works well for clothing that is topologically similar to the body. Besides, Neural Actor [5] requires multi-view input in order to obtain a complete texture map for network training. Note that we also use SMPL model in our approach; but we do not explicitly depend on the SMPL topology for shape and appearance representation. Therefore, our method is more general than Neural Actor [5] in terms of the cloth topology, and can work with partial input such as a monocular video.

DDC [2] is another state-of-the-art method for building animatable avatars. It demonstrates impressive results for loose clothes and even achieves real-time rendering performance. However, DDC requires a pre-scanned template model of the actor; that is why we do not compare with it since person-specific templates are not available in our experiment setting. In contrast, our method can model the dynamic shape and appearance of general garments without any pre-scanning efforts.

Some methods like **TNR** [11] and **ANR** [10] learn animatable avatars in 2D domain. They typically define appearance features (RGB color values or high-dimensional features) on the UV map of a body template, and exploit a 2D convolutional network to obtain the final color image. These methods not only suffer from the same limitation as [5, 8], but also fail to guarantee view consistency when rendering free-viewpoint images. Our method focuses more on creating a **3D** model, thereby significantly departing from this line of works.

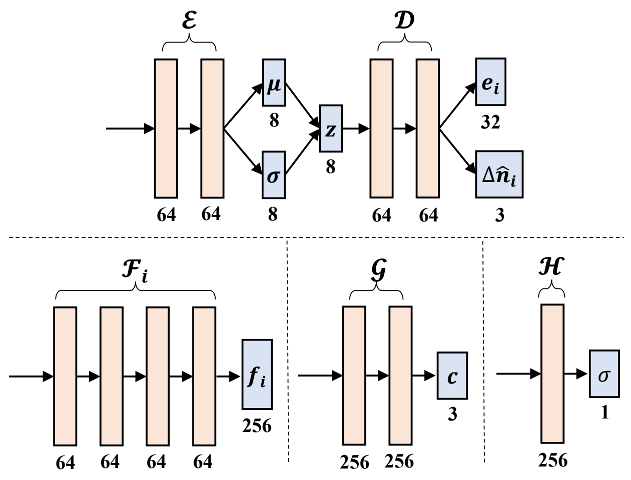


Figure A. **Architecture of our network.** Each orange rectangle represents a fully-connected layer followed by ReLU activation, and the numbers of output channels are labeled underneath.

MVP [6] also proposes to use local volumetric representation for deformable surface rendering. However, our work is essentially different from MVP: 1) MVP requires an estimate of scene geometry to construct the volumetric primitives, while our method works without knowing scene geometry; 2) MVP assumes accurate tracking of scene geometry over time, while our method is carefully designed to directly learn the motion hierarchy from data; 3) MVP only handles head movements and facial expressions, while our method can deal with challenging body motions and cloth deformations; 4) MVP mainly focuses on efficient rendering of training frames, while our method supports novel pose generation with explicit pose control.

C. Implementation Details

C.1. Architecture Details

We illustrate the network architecture in Fig. A. Note that before feeding the coordinates, view directions and time stamps into the MLP, we augment them using sinusoidal encoding, which is defined as:

$$\gamma(\mathbf{x}) = \big(\mathbf{x}, \sin(\mathbf{x}), \cos(\mathbf{x}), ..., \sin(2^{m-1}\mathbf{x}), \cos(2^{m-1}\mathbf{x})\big)\,.$$

The value of m is 6 for coordinates, 4 for view directions and 12 for time stamps. We normalize the time stamp before sinusoidal encoding, *e.g.*, the time stamp for t-th frame is normalized to t/T, where T is the total number of frames.

Note that the vanilla NeRF adopts a hierarchical sampling strategy and simultaneously optimizes two networks (one "coarse" and one "fine"), while we only train one network with uniform sampling for fair comparison against baseline methods.

Table A. Hyperparameters for network training and evaluation.

Parameter Name	Value
N (Number of Nodes)	128
σ (In Eqn. 6)	0.05
ϵ (In Eqn. 6)	0.001
λ_{rec} (In Eqn. 11)	1.0
λ_{trans} (In Eqn. 11)	0.02
λ_{ebd} (In Eqn. 11)	0.1
λ_{KL} (In Eqn. 11)	1×10^{-5}
Dimension of $\boldsymbol{e}_i$ (In Eqn. 5)	32
Dimension of $\boldsymbol{z}_i$ (In Eqn. 9)	8
Number of Ray Samples Per Batch	2048
Number of Point Samples Per Ray	64
Batch Size	4
Learning Rate	5×10^{-4}
Adam β_1	0.9
Adam β_2	0.999

C.2. Network Acceleration

Naively implementing our network will lead to heavy computational complexity, as one needs to query every local network for all point samples. To reduce network queries and accelerate program execution, we exploit the fact that for any point in the posed space, only a small portion of nodes have an influence on its color and density value. This is because the influence range of the nodes is truncated, as mathematically defined in Eqn. 6. Based on this observation, we implement custom CUDA kernels for acceleration purpose. To be more specific, let S denote the number of point samples and N the number of nodes (which is also the number of local MLPs). In the naive implementation, the points are first transformed into the local coordinate systems of the nodes, which results into a tensor of size $N \times S \times 3$ being fed into the network. In our optimized implementation, we first calculate the number of necessary point queries for each local MLP (indexed by i), which is denoted as S_i. Then we construct an empty tensor of size $N \times S' \times 3$, where $S' = \max\{S_1, S_2, ..., S_N\}$. By investigating the values of blending weights, we pick the valid elements in the original tensor and rearrange them into the new one, which is finally fed into the network. With our optimized implementation, the memory consumption decreases about 85%, and the running time decreases by a factor of 4.

D. Experimental Details

D.1. Dataset

In our experiments, we mainly use the following dataset:

- Dataset from [2]. We use two dress sequences ("Ling" and "FranziBlue") in this dataset. Each sequence contains about 20000 training frames captured using 100 cameras, but we manually select 20 views among them for computational efficiency.
- Dataset from [3]. We use one sweater sequence ("Lan") in this dataset, which is captured from 11 cameras and contains about 30000 training frames.

Table B. **Quantitative evaluation of the node-related variables.** We generate the images for training poses under different settings and report the averaged PSNR scores of all frames.

Setting	w/o $\{\Delta \boldsymbol{n}_i^{(t)}\}$ or $\{\boldsymbol{e}_i^{(t)}\}$	w/o $\{\boldsymbol{e}_i^{(t)}\}$	Full
Corresponding figure	Fig. 3 (b)	Fig. 3 (c)	Fig. 3 (d)
PSNR	17.52	20.47	21.48

Table C. **Quantitative evaluation of our cVAE design.** We replace the cVAE with a determinstic regression network and report the reconstruction accuracy (PSNR) of training frames.

Setting	Deterministic	Ours
Corresponding figure	Fig. 9 (b,e)	Fig. 9 (c,f)
PSNR	25.34	25.62

- ZJU-MoCap dataset [9]. We mainly conduct experiments on two sequences ("CoreView387" and "CoreView392"). Each sequence contains about 300 training frames captured from 23 view points, but we only use 4 view points among them for fair comparison against Neural Body [9].
- Multi-view dataset collected by ourselves. We built up a multi-view system that consists of 24 uniformly distributed cameras. Our system can capture synchronized videos at 30Hz with a resolution of 1024×1024. We collect data for three subjects and the frame numbers of videos range from 2500 to 5000.

We use [12] to register SMPL(-X) model to the video frames, and use BackgroundMattingV2 [4] for foreground segmentation.

D.2. Training Details

We use PyTorch to implement our networks. The hyperparameters needed for network implementation and training are reported in Tab. A. Note that during network training, the learning rate decays exponentially every 20k iterations. The number of iterations is set to 100k for People Snapshot dataset [1], 300k for ZJU-MoCap dataset [9] and 500k for other multi-view sequences. For baseline methods, we use the author-provided code and run all the experiments using the default training settings.

D.3. Metrics

As described in the paper, we use two standard metrics, peak signal-to-noise ratio (PSNR) and structural similarity index (SSIM), for quantitative evaluation. To reduce the influence of background pixels, all the scores are calculated from the images cropped with a 2D bounding box which is estimated from the projection of SMPL model. More details are described in this link.

E. More Experiments

Quantitative ablation of node-related variables. We conduct a qualitative ablation study on the effects of the node-related variables in Fig. 3 in the main paper. In Tab. B we report the corresponding quantitative results across all frames to further evaluate the impact of the node-related variables.

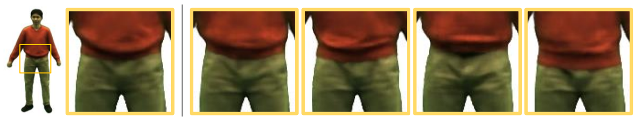


Figure B. **Impact of the latent $\boldsymbol{z}_i$.** We show the testing results when the same pose are given but $\boldsymbol{z}_i$ is set to zero (leftmost) or assigned with random Gaussian noises (right).

Quantitative ablation of our cVAE design. Similarly, we report the corresponding quantitative results across all frames in Tab. C to further evaluate our cVAE design. The numeric results further prove that our cVAE design is critical for better reconstructing the realistic details in the training frames, which is consistent with our conclusion in the main paper.

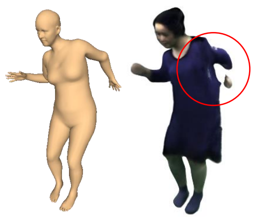
cVAE ablation with novel poses. In Fig. 9 we mainly conduct the cVAE ablation study on training frames. In the right inset figure we conduct an identical experiment using novel poses from a testing sequence. The results also show that our cVAE design is benefitial for learning sharper wrinkle details.

w/o cVAE
w/ cVAE

Impact of the latent $\boldsymbol{z}_i$. As we mentioned in Sec. 4.1, we set $\boldsymbol{z}_i$ to zeros when synthesizing images of novel poses. In fact, latent $\boldsymbol{z}_i$ does not have to be zeros and can be modified in accordance of applications. In Fig. B, we show that modifying the latent $\boldsymbol{z}_i$ will lead to different wrinkle patterns. This feature can be further explored to generate multiple plausible animation sequences, and we leave it as future work.

F. Limitation and Future Work

As we discuss in Sec.6 in the main paper, our method may fail to generate plausible results when the animation poses starkly differ from the training poses; see the inset figure on the right for an example. The main reason for this phenomenon is that neither subject-specific templates nor the SMPL surface is used to regularize shape learning in our method. Consequently, we cannot guarantee that our model is fully aware of the underlying geometry and its articulated surface deformation. Geometrical priors of clothed humans [7] can be employed to resolve this limitation and we leave it for future work.



In addition, the dynamic deformations and wrinkle changes of garments involve complex physics processes, which may be beyond the representation capability of sparse nodes. Denser nodes could probably alleviate this limitation, but this will result in heavier computational burden. In fact, modeling the physics attributes of real-world garments is a long-standing, extremely difficult problem in computer graphics. We are currently seeking a better approach that can combine the merits of learning-based implicit representations and physics-based cloth simulation.

References

[1] Thiemo Alldieck, Marcus Magnor, Weipeng Xu, Christian Theobalt, and Gerard Pons-Moll. Video based reconstruction of 3d people models. In *CVPR*, 2018. 3

[2] Marc Habermann, Lingjie Liu, Weipeng Xu, Michael Zollhoefer, Gerard Pons-Moll, and Christian Theobalt. Real-time deep dynamic characters. *ACM TOG*, 40(4), aug 2021. 1, 2

[3] Marc Habermann, Weipeng Xu, Michael Zollhofer, Gerard Pons-Moll, and Christian Theobalt. Deepcap: Monocular human performance capture using weak supervision. In *CVPR*, 2020. 2

[4] Shanchuan Lin, Andrey Ryabtsev, Soumyadip Sengupta, Brian L. Curless, Steven M. Seitz, and Ira Kemelmacher-Shlizerman. Real-time high-resolution background matting. In *CVPR*, 2021. 3

[5] Lingjie Liu, Marc Habermann, Viktor Rudnev, Kripasindhu Sarkar, Jiatao Gu, and Christian Theobalt. Neural actor: Neural free-view synthesis of human actors with pose control. *ACM TOG (ACM SIGGRAPH Asia)*, 2021. 1

[6] Stephen Lombardi, Tomas Simon, Gabriel Schwartz, Michael Zollhöfer, Yaser Sheikh, and Jason M. Saragih. Mixture of volumetric primitives for efficient neural rendering. *ACM TOG*, 40(4):59:1–59:13, 2021. 2

[7] Qianli Ma, Jinlong Yang, Siyu Tang, and Michael J. Black. The power of points for modeling humans in clothing. In *ICCV*, 2021. 3

[8] 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*, 2021. 1

[9] 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, 3

[10] Amit Raj, Julian Tanke, James Hays, Minh Vo, Carsten Stoll, and Christoph Lassner. ANR: articulated neural rendering for virtual avatars. In *CVPR*, 2021. 1

[11] Aliaksandra Shysheya, Egor Zakharov, Kara-Ali Aliev, Renat Bashirov, Egor Burkov, Karim Iskakov, Aleksei Ivakhnenko, Yury Malkov, Igor Pasechnik, Dmitry Ulyanov, Alexander Vakhitov, and Victor Lempitsky. Textured neural avatars. In *CVPR*, 2019. 1

[12] Yuxiang Zhang, Zhe Li, Liang An, Mengcheng Li, Tao Yu, and Yebin Liu. Light-weight multi-person total capture using sparse multi-view cameras. In *ICCV*, 2021. 3