Supplementary Material for AvatarCap: Animatable Avatar Conditioned Monocular Human Volumetric Capture

Zhe Li, Zerong Zheng, Hongwen Zhang, Chaonan Ji, and Yebin Liu

Department of Automation, Tsinghua University, China

In this supplementary material, we provide the implementation details of our method and more experiments. Please see the supplementary video for more visualization of our results.

1 Implementation Details

1.1 Data Collection and Preprocessing

The textured scans are captured using a dense DLSR rig as the training database for creating the avatar as shown in Fig. 1. We firstly fit SMPL [6] to each scan using [1]. Then the scan is deformed to the canonical pose following ARCH [2]. Different from directly learning SDF from the non-watertight canonicalized scans [13], we non-rigidly deform a canonical SMPL to align with the scan for filling the holes, then utilize Poisson reconstruction [3] to generate watertight scans. Finally, to jointly train the texture template represented by NeRF [9], we render the original textured scans from 60 views distributed uniformly in a circle.

Note that the texture and occupancy supervisions are not in the same space, i.e., the former is in the posed space while the later is in the canonical space. The reason for that is there may exist body part intersections on the original captured scans, e.g., the armpits, if we sample points around these regions in the posed space, the corresponding ground-truth occupancy values will be incorrect.

1.2 GeoTexAvatar

Network Architecture. The GeoTexAvatar network contains two modules, i.e., the Geo-Tex implicit template and the pose-conditioned warping field. The Geo-Tex implicit template is represented as an MLP, which takes a 3D template point with 10th-order positional encoding [9, 15] as input, and returns its occupancy, color and density. The template network consists of a shared MLP with (63, 256, 256, 256, 256, 256, 256, 256) neurons, a geometry MLP with (256, 128, 2) neurons and a color MLP with (256, 256, 128, 3) neurons. The geometry MLP jointly outputs the occupancy and density value; such an implicit representation is inspired by [14]. The last non-linear activation functions of occupancy, density and color MLPs are Sigmoid, ReLU and Sigmoid, respectively.

The pose-conditioned warping field consists of a positional map encoder $E(\cdot)$ and an offset decoder $D(\cdot)$:

$$\Delta W(\mathbf{x}_c, \boldsymbol{\theta}) = D(\mathbf{x}_c, B(\pi(\mathbf{x}_c); E(\mathbf{P}(\boldsymbol{\theta})))), \tag{1}$$

Fig. 1: **Training scans of one subject.**

where $\mathbf{x}_c$ is a canonical 3D point, $\mathbf{P}(\boldsymbol{\theta})$ is the rendered canonical SMPL positional map where the pixel value is the posed SMPL vertex position, $B(\cdot)$ is a bilinear sampling function to sample feature on the feature map $E(\mathbf{P}(\boldsymbol{\theta}))$ for $\mathbf{x}_c$, and $\pi(\cdot)$ is the orthographic projection to project $\mathbf{x}_c$ onto the 2D plane of the rendered positional map. To generate the positional map, we render the canonical SMPL from front and back views to generate two pixel-aligned positional maps, then concatenate them together, and finally feed them to the positional map encoder $E(\cdot)$ followed by the offset decoder $D(\cdot)$. Following [8], the positional map encoder $E(\cdot)$ is a UNet [10] that contains seven [Conv2d, BatchNorm, LeakyReLU(0.2)] blocks, followed by seven [ReLU, ConvTranspose2d, BatchNorm] blocks, and it returns a $256 \times 256 \times 64$ feature map. The offset decoder is a MLP that takes the canonical point and corresponding feature as input, and it contains (3+64, 256, 256, 256, 256, 256, 256, 256, 3) neurons at each layer, respectively. Note that in [8] the SMPL positional map in defined in the SMPL UV space. We do not follow the practice in [8] because we need to query the feature for the whole 3D space. Our definition also avoids the discontinuity in the UV space which causes the seam artifacts on the back of animated models in [8].

Training. We train the whole network in an end-to-end manner using the Adam [4] optimizer with a batch size of 4 for 30 epochs on ~ 20 scans. The loss weights are set as $\lambda_{\text{geo}} = 0.5, \lambda_{\text{tex}} = 1.0, \lambda_{\text{reg}} = 0.1$. The initial learning rates of the Geo-Tex implicit template and warping field are 1×10^{-3} and 1×10^{-4}, respectively, and drop half every 20000 iterations. We initialize the warping field to output zero offsets, and at the first two epochs, we fix the warping field and only optimize the template network to obtain a coarse template. The training of one subject for creating an animatable avatar takes about two hours.

1.3 Avatar-conditioned Volumetric Capture

As shown in Fig. 3 of the main paper, the initialization of the volumetric capture contains avatar animation and normal map canonicalization.

Avatar Animation. Firstly, we can estimate SMPL pose from the monocular color input using SPIN [5] or PyMAF [19]. With the SMPL pose, we can generate canonical SMPL positional map as described in Sec. 1.2. We allocate a canonical volume that contains the canonical SMPL body. For each voxel, we feed its position and projected feature on the convoluted feature map to the network to evaluate its occupancy, then we perform Marching Cubes [7] on this occupancy volume to acquire a canonical geometric model. Finally, we render it from front and back views by orthographic projection to obtain front and back avatar normal maps.

Normal Map Canonicalization. In this branch, we firstly estimate the normal map from the monocular color input using pix2pixHD [17] following PIFuHD [12]. Then we deform the canonical avatar model using the estimated SMPL pose to the image/posed space, then project it onto the normal map to fetch a normal vector for each visible vertex. Similar to avatar animation, we render the fetched normals using the canonical avatar from the same front and back views by orthographic projection to obtain front and back image-observed normal maps.

With the above two steps in the initialization, we bridge the avatar and image information on the unified 2D canonical image plane.

Canonical Normal Fusion As introduced in Sec. 5.1 of the main paper, we formulate the fusion as an optimization, and in the energy function Eq. 6, we set $\lambda_{\text{fitting}} = 1.0$ and $\lambda_{\text{smooth}} = 1.0$, and we optimize it using Gauss-Newton algorithm for 50 iterations. The resolution of all the normal maps are 512×512, and the resolution of the rotation grids is 64×64.

Model Reconstruction We introduce a reconstruction network pretrained on a large-scale human dataset (THuman 2.0 [18]) to leverage the data prior to infer the 3D model from the fused normal maps. Because the normal maps are in the canonical space, similar to Sec. 1.1, we deform all the original scans to the canonical pose by the SMPL registration. Then we render the canonicalized scan from front and back views to obtain normal maps. We sample 3D points randomly near surfaces and in the canonical volume as in PIFu [11], then calculate their occupancy values. With the rendered normal maps and sampled points, we train this network using the Adam [4] optimizer with a batch size of 4 and a learning rate of 1×10^{-3} for 240 epochs. The training takes about two days on one RTX 3090 GPU.

1.4 Runtime Performance

Given $\sim$20 textured scans of one subject, it takes about 0.5 hours for the data preprocessing and 2.0 hours for the avatar training. In the volumetric capture, the avatar animation, normal map canonicalization, canonical normal fusion, model reconstruction and texture generation cost about 1.0, 0.5, 1.2, 0.8, 3.0 secs, respectively. Overall, our method takes about $6 \sim 7$ secs for reconstructing one frame.

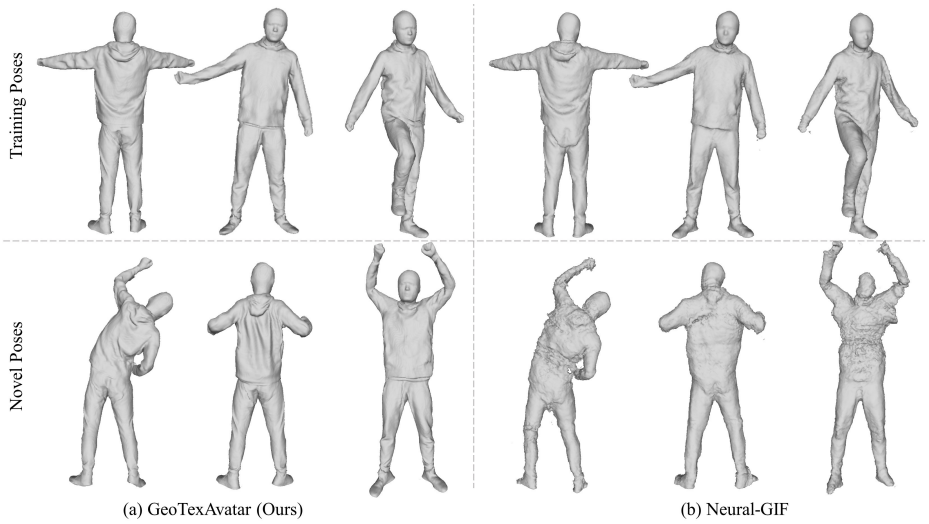

Fig. 2: **Comparison between GeoTexAvatar and Neural-GIF [16].** We show animated results by our GeoTexAvatar and Neural-GIF on both training and novel poses, respectively.

2 Additional Experiment

Comparison against Neural-GIF [16]. We further compare our animatable avatar module, GeoTexAvatar, against another state-of-the-art scan-based avatar method, Neural-GIF [16]. Fig. 2 shows the animated results of our method and Neural-GIF on both training and novel poses, respectively. It shows that Neural-GIF suffers from overfitting, and cannot generalize the avatar trained on 22 scans to the novel poses. We hypothesize that the reasons include: 1) It is hard for the inverse skinning network in Neural-GIF to learn good generalization from only few examples, because its input coordinate is in the posed space where the skinning weight of the same position varies significantly when the SMPL pose changes; 2) Neural-GIF does not decompose the pose-agnostic details and the pose-dependent ones and it conditions both the displacement and canonical SDF networks on the pose input, thus all the surface details are driven by the pose input. Benefiting from the decomposition between pose-agnostic and pose-dependent details, our method realizes more robust and plausible pose generalization.

References

1. https://github.com/zju3dv/EasyMocap
2. Huang, Z., Xu, Y., Lassner, C., Li, H., Tung, T.: Arch: Animatable reconstruction of clothed humans. In: CVPR. pp. 3093–3102 (2020)
3. Kazhdan, M., Bolitho, M., Hoppe, H.: Poisson surface reconstruction. In: Proceedings of the fourth Eurographics symposium on Geometry processing. vol. 7 (2006)
4. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. In: ICLR (2015)
5. Kolotouros, N., Pavlakos, G., Black, M.J., Daniilidis, K.: Learning to reconstruct 3d human pose and shape via model-fitting in the loop. In: ICCV. pp. 2252–2261 (2019)
6. Loper, M., Mahmood, N., Romero, J., Pons-Moll, G., Black, M.J.: Smpl: A skinned multi-person linear model. TOG **34**(6), 1–16 (2015)
7. Lorensen, W.E., Cline, H.E.: Marching cubes: A high resolution 3d surface construction algorithm. TOG **21**(4), 163–169 (1987)
8. Ma, Q., Yang, J., Tang, S., Black, M.J.: The power of points for modeling humans in clothing. In: ICCV. pp. 10974–10984 (2021)
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: ECCV. pp. 405–421. Springer (2020)
10. Ronneberger, O., Fischer, P., Brox, T.: U-net: Convolutional networks for biomedical image segmentation. In: International Conference on Medical image computing and computer-assisted intervention. pp. 234–241. Springer (2015)
11. Saito, S., Huang, Z., Natsume, R., Morishima, S., Kanazawa, A., Li, H.: Pifu: Pixel-aligned implicit function for high-resolution clothed human digitization. In: ICCV. pp. 2304–2314 (2019)
12. Saito, S., Simon, T., Saragih, J., Joo, H.: Pifuhd: Multi-level pixel-aligned implicit function for high-resolution 3d human digitization. In: CVPR (June 2020)
13. Saito, S., Yang, J., Ma, Q., Black, M.J.: Scanimate: Weakly supervised learning of skinned clothed avatar networks. In: CVPR. pp. 2886–2897 (2021)
14. Shao, R., Zhang, H., Zhang, H., Chen, M., Cao, Y., Yu, T., Liu, Y.: Doublefield: Bridging the neural surface and radiance fields for high-fidelity human reconstruction and rendering. In: CVPR (2022)
15. Tancik, M., Srinivasan, P., Mildenhall, B., Fridovich-Keil, S., Raghavan, N., Singhal, U., Ramamoorthi, R., Barron, J., Ng, R.: Fourier features let networks learn high frequency functions in low dimensional domains. Advances in Neural Information Processing Systems **33**, 7537–7547 (2020)
16. Tiwari, G., Sarafianos, N., Tung, T., Pons-Moll, G.: Neural-gif: Neural generalized implicit functions for animating people in clothing. In: ICCV. pp. 11708–11718 (2021)
17. Wang, T.C., Liu, M.Y., Zhu, J.Y., Tao, A., Kautz, J., Catanzaro, B.: High-resolution image synthesis and semantic manipulation with conditional gans. In: CVPR. pp. 8798–8807 (2018)
18. Yu, T., Zheng, Z., Guo, K., Liu, P., Dai, Q., Liu, Y.: Function4d: Real-time human volumetric capture from very sparse consumer rgbd sensors. In: CVPR. pp. 5746–5756 (2021)
19. Zhang, H., Tian, Y., Zhou, X., Ouyang, W., Liu, Y., Wang, L., Sun, Z.: Pymaf: 3d human pose and shape regression with pyramidal mesh alignment feedback loop. In: ICCV. pp. 11446–11456 (2021)