

PointAvatar: Deformable Point-based Head Avatars from Videos

Yufeng Zheng^{1,2} Wang Yifan³ Gordon Wetzstein³ Michael J. Black² Otmar Hilliges¹
¹ETH Zurich, ²Max Planck Institute for Intelligent Systems, ³Stanford University

In this supplemental document, we provide the additional experiments mentioned in the main paper in Sec. 1, implementation details for our method in Sec. 2, the proof for Jacobian-based normal transformation in Sec. 3, and discussions about data capture ethics in Sec. 4.

Additionally, we strongly recommend checking out (1) our supplemental video, and (2) examples of learned 3D point clouds.

1. Additional Results

1.1. Results on the MakeHuman Dataset

We evaluate the reconstructed surface geometry on the MakeHuman synthetic dataset rendered from [1]. We show that PointAvatar is not only capable of capturing volumetric structures as shown in the main paper, but also performs on par with state-of-the-art methods on surface geometry reconstruction, as shown in Fig. 1 and Tab. 1.

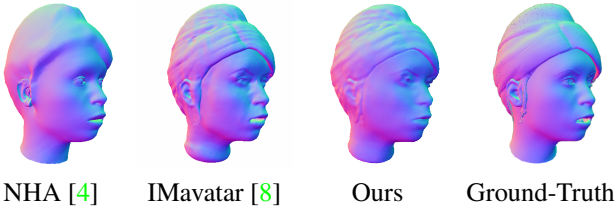


Figure 1. **Qualitative comparison of surface geometry.** Our point-based avatar representation reconstructs comparable head geometry to surface-only representations such as meshes (NHA) and implicit surfaces (IMavatar).

Method	female 1	female 2	male 1	male 2
NHA [4]	0.94	0.95	0.94	0.94
IMavatar [8]	0.961	0.966	0.954	0.955
Ours	0.954	0.954	0.944	0.958

Table 1. **Quantitative comparison on the MakeHuman dataset [1].** We report the normal consistency metric for evaluating reconstructed surface geometry. The scores for NHA [4] and IMavatar [8] are obtained from their papers.

We thank the authors of NHA [4] for kindly sharing this evaluation dataset with us.

1.2. Learning without Foreground Masks

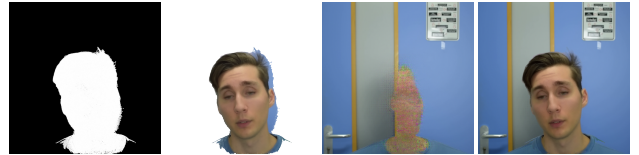


Figure 2. **Self-supervised foreground-background disentanglement.** PointAvatar can be learned without mask supervision or known background.

It is beneficial to learn avatars without relying on foreground segmentation masks, because off-the-shelf segmentation networks often fail in in-the-wild scenarios. In Fig. 2, we show that PointAvatar is able to roughly disentangle foreground and background contents in a self-supervised manner. Note that NerFace [3] cannot be learned without known backgrounds.

2. Implementation Details

In this section, we provide implementation details regarding network architectures, training and evaluation procedures, and the preprocessing of videos.

2.1. Network Architecture

We show the architecture of the canonical, deformation and shading MLPs in Fig. 3.

2.2. Training Details

Loss weights and optimization. We choose $\lambda_{\text{rgb}} = 1$, $\lambda_{\text{mask}} = 1$, $\lambda_{\text{flame}} = 1$, and $\lambda_{\text{vgg}} = 0.1$ for most of our experiments, except for the experiment shown in Fig. 6 (main paper), where we used $\lambda_{\text{rgb}} = 5$, and $\lambda_{\text{vgg}} = 0.05$, and the experiment on synthetic data (Sec. 1.1 in Supp. Mat.), where we used $\lambda_{\text{rgb}} = 10$, and $\lambda_{\text{vgg}} = 0$. For the flame loss, we choose $\lambda_e = 1000$, $\lambda_p = 1000$, and $\lambda_w = 1$. For optimizing the canonical SDF, we use $\lambda_{\text{SDF}} = 1$, and $\lambda_{\text{eik}} =$

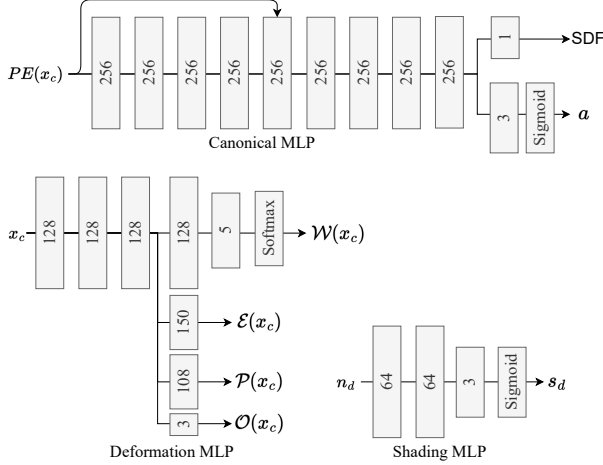


Figure 3. **Network architecture.** We show the network architectures of the canonical, deformation and shading MLPs. Except from the last layer, each linear layer is followed by weight normalization [7] and non-linear activation. We use the Softplus [2] function for the canonical and deformation MLP, and the ReLU activation for the shading MLP.

0.1. We train our models using an Adam optimizer [5] for 63 epochs, with a learning rate of $\eta = 1e^{-4}$, and $\beta = (0.9, 0.999)$.

Static bone transformation. In the pre-processing step, the FLAME tracking of the upper body is often unstable. We handle this problem via the introduction of an additional static bone, similar to IMavatar [8]. Since the upper body remains mostly static in the training video, the deformation MLP assigns large LBS weights to the static bone to reduce shoulder motion under different FLAME pose configurations. This trick effectively reduces shoulder jittering.

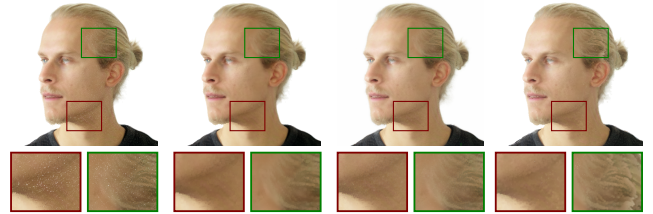
Point upsampling and pruning. Every 5 epochs, we double the number of points and reduce the point rasterization radii, which allows our method to capture finer details. The new point locations are obtained by perturbing the current points with random noises. The factor for radius reduction is carefully chosen: Assuming that points roughly form a surface, the radius should be decreased by a factor of $1/\sqrt{2}$. We choose to reduce the point radii by a factor of 0.75 in practice. Additionally, after each epoch, we prune away points that are invisible, *i.e.*, not projected onto any pixels with compositing weights larger than a threshold. We choose 0.5 as the pruning threshold.

2.3. Evaluation Details

FLAME [6] parameter optimization. Similar to NHA [4], our method also fine-tunes pre-tracked FLAME

expression and pose parameters during training and evaluation. For training, the loss weights are elaborated in the main paper and in Sec. 2.2 of Supp. Mat. For test-time tracking optimization, we only use the RGB loss.

Hole filling. Rendered images from PointAvatar may contain “white dot” artifacts. This happens due to the sparsity of deformed point clouds which reveals the white background. The issue mainly occurs around (1) stretched regions (*e.g.* neck), in this case, the deformation causes sparsity; (2) regions that are infrequently observed during training (*e.g.* profile views), because unseen points are pruned during training (see Sec. 2.2 of Supp. Mat. for pruning strategy). In Fig. 4, we compare several post-processing methods to address this artifact. (1) We apply erosion and dilation consecutively to the rendered images. This method fills most white dots but slightly blurs images. (2) We train a per-subject 2D neural network (NN) to remove white dots from the rendered images. This method robustly removes artifacts and does not blur images, but it requires cumbersome per-subject NN training. More specifically, we augment GT images with white dots using the rendered mask, and train a 2D NN to remove the added artifacts. We found the trained NN can remove artifacts from rendered images despite the domain gap, and also generalizes to unseen poses. (3) We increase point radii based on distances to neighbors before rasterization. This method principally tackles the sparsity problem in 3D space, but often falsely thickens thin structures such as hair strands. We use the first method in the paper due to its simplicity and effectiveness, but adaptive radii and neural networks could also be leveraged for better performance in some cases.



orig. RGB (1) erode-dilate (2) 2D NN (3) adapt radii
Figure 4. **Point sparsity and white dot artifacts.** We show the rendered image which contains white dot artifacts (orig. RGB), and refined results from different post-processing methods.

Relighting. Our method learns to disentangle *albedo* from pose-dependent lighting effects which we refer to as *shading*. Previous works that use MLP-based texture networks transform 3D locations and normal directions into shaded colors. In contrast, we perform disentanglement by conditioning the albedo only on the canonical locations and the shading only on the normal directions. This is how we achieve *unsupervised* albedo disentanglement.

We found our method allows for rudimentary relighting (1) by manipulating the normal directions of the shading component, which changes the light direction (please check the teaser and Fig. 5 in the main paper for horizontally-flipped and rotated light directions), or (2) by discarding the learned shading and instead rendering with a default shading model using the learned albedo and normals. The latter option allows for relighting in novel scenes. In the supplementary video (03:39), we use a diffused shading model to relight avatars, but it leads to less faithful appearances compared to the learned shading.

2.4. Data Preprocessing

We leverage the preprocessing pipeline of IMavatar [8] and use the same camera and FLAME parameters for all SOTA methods. This allows us to compare head avatar methods without considering the influence of different face-tracking schemes used for preprocessing.

3. Proof of Eq. 6 (deformed normal formula)

First, we recap Eq. 6 from the main paper, which states that deformed normals can be obtained as:

$$\mathbf{n}_d = l \mathbf{n}_c \left(\frac{d\mathbf{x}_d}{d\mathbf{x}_c} \right)^{-1}$$

To prove it, we assume points form flat surfaces locally. The deformed normal \mathbf{n}_d of a point can then be defined via the constraint:

$$\mathbf{n}_d \cdot (\mathbf{x}_{d,i} - \mathbf{x}_d) = 0,$$

which needs to be satisfied for every neighboring point $\mathbf{x}_{d,i}$. Note that $\mathbf{x}_{d,i}$ can be linearly approximated as

$$\mathbf{x}_{d,i} = \mathbf{x}_d + \frac{d\mathbf{x}_d}{d\mathbf{x}_c}(\mathbf{x}_{c,i} - \mathbf{x}_c),$$

which allows us to rewrite the constraint as

$$\mathbf{n}_d \cdot \frac{d\mathbf{x}_d}{d\mathbf{x}_c}(\mathbf{x}_{c,i} - \mathbf{x}_c) = 0.$$

Note that the canonical normal \mathbf{n}_c satisfies

$$\mathbf{n}_c \cdot (\mathbf{x}_{c,i} - \mathbf{x}_c) = 0.$$

Therefore, in order to satisfy the constraint for all neighboring points $\mathbf{x}_{d,i}$, \mathbf{n}_d must satisfy

$$\mathbf{n}_d \cdot \frac{d\mathbf{x}_d}{d\mathbf{x}_c} = l \mathbf{n}_c,$$

where l is a normalizing scalar to ensure unit normal length. This leads to the formulation in Eq. 6.

4. Ethics

We captured 5 human subjects with smartphones or laptop cameras for our experiments. All subjects have given written consents for using the captured images in this project. We will make the data publicly available for research purposes, where permission for data publishing is given by the subjects.

Our method could be extended to generate media content of real people performing synthetic poses and expressions. We do not condone using our work to generate fake images or videos of any person with the intent of spreading misinformation or tarnishing their reputation.

References

- [1] Leyde Briceno and Gunther Paul. Makehuman: a review of the modelling framework. In *Proceedings of the 20th Congress of the International Ergonomics Association (IEA 2018) Volume V: Human Simulation and Virtual Environments, Work With Computing Systems (WWCS), Process Control 20*, pages 224–232. Springer, 2019. 1
- [2] Charles Dugas, Yoshua Bengio, François Bélisle, Claude Nadeau, and René Garcia. Incorporating second-order functional knowledge for better option pricing. In T. Leen, T. Dietterich, and V. Tresp, editors, *Advances in Neural Information Processing Systems*, volume 13. MIT Press, 2001. 2
- [3] Guy Gafni, Justus Thies, Michael Zollhöfer, and Matthias Nießner. Dynamic neural radiance fields for monocular 4d facial avatar reconstruction. In *Computer Vision and Pattern Recognition (CVPR)*, pages 8649–8658, June 2021. 1
- [4] Philip-William Grassal, Malte Prinzler, Titus Leistner, Carsten Rother, Matthias Nießner, and Justus Thies. Neural head avatars from monocular rgb videos. In *Computer Vision and Pattern Recognition (CVPR)*, 2022. 1, 2
- [5] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 2
- [6] Tianye Li, Timo Bolkart, Michael J. Black, Hao Li, and Javier Romero. Learning a model of facial shape and expression from 4D scans. *ACM Transactions on Graphics, (Proc. SIGGRAPH Asia)*, 36(6):194:1–194:17, 2017. 2
- [7] Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, Xi Chen, and Xi Chen. Improved techniques for training gans. In D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc., 2016. 2
- [8] Yufeng Zheng, Victoria Fernández Abrevaya, Marcel C. Bühler, Xu Chen, Michael J. Black, and Otmar Hilliges. I M Avatar: Implicit morphable head avatars from videos. In *Computer Vision and Pattern Recognition (CVPR)*, 2022. 1, 2, 3