Realistic One-shot Mesh-based Head Avatars

Taras Khakhulin[1,2], Vanessa Sklyarova[1,2],
Victor Lempitsky[3], and Egor Zakharov[1,2]

[1] Samsung AI Center – Moscow
[2] Skolkovo Institute of Science and Technology
[3] Yandex Armenia

https://samsunglabs.github.io/rome/

1 Supplementary material

1.1 Implementation details

Photometric training objectives. During training, we use the photometric loss $\mathcal{L}_{\text{photo}}$ to aid in learning the geometry, as well as to train the rendering. Our photometric loss is the combination of the perceptual, the identity, the adversarial, and the segmentation losses:

$$\mathcal{L}_{\text{photo}} = \lambda_{\text{per}}\mathcal{L}_{\text{per}} + \lambda_{\text{idt}}\mathcal{L}_{\text{idt}} + \lambda_{\text{adv}}\mathcal{L}_{\text{adv}} + \lambda_{\text{seg}}\mathcal{L}_{\text{seg}}. \tag{1}$$

For the perceptual loss $\mathcal{L}_{\text{per}}$, we use a weighted combination of distances between features of predicted and target images. These features are taken from two pre-trained convolutional neural networks. We use features from a VGG19 [10] network pre-trained on ImageNet [9] to match the general content and features from a VGG16-based gaze detection network [2] to match the gaze direction. This loss, therefore, can be expressed as follows:

$$\mathcal{L}_{\text{per}} = \mathcal{L}_{\text{vgg}} + \mathcal{L}_{\text{gaze}}. \tag{2}$$

In both of these losses, we measure the L1 distance between conv1_1, conv2_1, conv3_1, conv4_1, and conv5_1 features after ReLU activations. We then sum these distances with the following weights: $\frac{1}{32}$, $\frac{1}{16}$, $\frac{1}{8}$, $\frac{1}{4}$, and 1 to obtain the loss value.

In the gaze detection network, we independently process both eyes and average the losses corresponding to each one of them. Before feature extraction, we additionally perform alignment of both eyes using the keypoints extracted from the image via a differentiable interpolation. For more details, please refer to [2].

The face identity loss $\mathcal{L}_{\text{idt}}$ is the cosine distance between the embeddings of a pre-trained face recognition network [1], evaluated for the predicted and the target images. We use this loss to better preserve the person's identity in the renders. We calculate the cosine distance as minus the cosine similarity.

To calculate the adversarial loss $\mathcal{L}_{\text{adv}}$ we jointly train a discriminator network alongside our reconstruction and rendering networks. We use a weighted

combination of the hinge-loss discriminator loss as well as the feature-matching loss [13]. We additionally apply spectral normalization [7] to the discriminator network. The configuration of the adversarial training closely follows the related works on image-to-image translation [12]. In particular, we use the PatchGAN [5] architecture for the discriminator.

Lastly, we use Dice loss to match the predicted segmentation $\hat{\mathbf{s}}_t$ to the ground-truth $\mathbf{s}_t$:

$$\mathcal{L}_{\text{seg}} = 1 - 2\frac{\hat{\mathbf{s}}_t \cdot \mathbf{s}_t}{\|\hat{\mathbf{s}}_t\|_2^2 + \|\mathbf{s}_t\|_2^2}, \tag{3}$$

where $\cdot$ denotes a scalar product. The segmentation loss is calculated using each batch element separately and then averaged across the batch.

Architectures of the neural networks. We use group normalization [14] paired with weight standardization [8] in all networks to facilitate training with smaller batch sizes. Empirically we found this combination to perform better than standard instance normalization [11].

In the autoencoder E_{tex}, which encodes the input image $\mathbf{x}_s$ into the neural texture $\mathbf{T}_s$, we use pre-activation residual blocks [4]. We set the number of channels in the neural texture $\mathbf{T}_s$ to eight (we observed that values between 8 and 16 result in similar performance). We additionally align the input image using the transformation similar to the one used in the FFHQ dataset [6]. We only modify the zoom-out factor to 1.25 so that the aligned image contains more hair and upper-body regions.

For the networks E_{img} and E_{geom} we use a standard U-Net architecture. The network E_{img} is U-Net, which predict an output image and the segmentation mask. Besides the neural texture, we additionally condition these two networks on the rendered mesh normals. Specifically, we process each input dimension of the normal vectors using $\{\sin(kx)\}_{k=1}^{K}$ and $\{\cos(kx)\}_{k=1}^{K}$ functions. In our experiments, we set $K = 6$. We then concatenate the resulting encodings to the neural texture. We use max-pooling layers for downsampling and nearest neighbors upsampling in the network, which decodes an image, and average pooling with bilinear upsampling in the network, which decodes segmentations. Additionally, the segmentation U-Net network has two times fewer channels than the image network.

The architecture of E_{geom} is the same as the image encoding U-Net, albeit with a different number of input and output channels. We set the dimensionality of a latent geometry map $\mathbf{Z}_t$, which is an output of E_{geom}, to 32. Additionally, we encode the xyz-texture using harmonic functions via the same process described before. We then concatenate the obtained embeddings to the initial xyz-texture. The resulting feature map has $3 + 3 \cdot 6 \cdot 2 = 39$ channels. In total, E_{geom} has $8 + 39 = 47$ input channels, since we also concatenate the embeddings of the xyz-texture to the neural texture.

Finally, G_{geom} consists of an MLP network which we apply separately for each vertex to predict its offsets. To obtain its inputs, we first resample the latent geometry map $\mathbf{Z}_t$ using an irregular grid specified by the texture coordinates

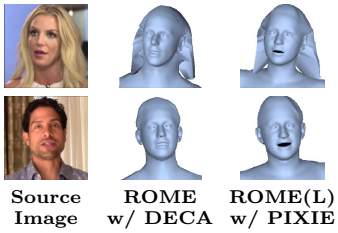
w. The obtained features for each vertex are denoted as $\mathbf{z}_t^w$. These features are concatenated to the harmonic embeddings of w in the same way as the xyz-texture. Specifically, each vector w has two dimensions, therefore we encode it into $2 + 2 \cdot 6 \cdot 2 = 26$ features, and then concatenate with 32 channels of $\mathbf{z}_t^w$ to obtain a vector with the dimensionality of 58 — the number of G_{geom} input dimensions.

Training details. We use the ADAM optimizer to train all networks in our model. We set the learning rate to $1 \cdot 10^{-4}$ for all the networks except for the discriminator. For it, we set the learning rate to $4 \cdot 10^{-4}$. We also set $\beta_1 = 0$, and $\beta_2 = 0.999$. We train using eight P40 NVIDIA GPUs to facilitate the batch size of 32. We observe that decreasing the batch size leads to visible degradation of both rendering and reconstructions, but we did no ablations to measure this effect.

1.2 Evaluation

We present additional results for the 3D reconstruction in both self-driving (reconstruction using frames from the same video), and cross-driving (reconstruction from a photo of one person and animation from a video of a different person) scenarios. In Figure 2, we present more self-driving results on the H3DS dataset, as well as a side-by-side comparison with H3D-Net. In Figure 3, we present cross-driving evaluation results using the hold-out samples from the VoxCeleb2 dataset, as well as paintings, the latter allowing us to evaluate the susceptibility of our approach to domain shifts.

(a) Comparison with PIFuHD.



(b) ROME integrated with PIXIE.

Additionally, we show the results in Fig. 1a. PIFuHD can only recover head geometry from the full-body images, which prevents us from doing a comparison with this method using existing head reconstruction and reenactment benchmarks. While having more detailed reconstructions, PIFuHD requires training with 3D supervision and does not have animation capabilities. Currently, these limitations can be lifted only at the cost of lengthy multi-shot training per each avatar (IMAvatars), or by sacrificing some detalization of reconstructions and retaining both real-time and one-shot capabilities, which is done in our method.

Method	LPIPS↓	SSIM↑	PSNR↑
w/o $\Delta\hat{\mathbf{v}}$	0.10	0.81	23.1
ROME	0.08	0.86	25.8

Table 1: Quantitative ablation on a hold-out set of the VoxCeleb2 dataset. We observe that the deferred neural rendering trained without offsets achieves lower image quality for face and hair regions, than a full ROME system.

To demonstrate the the ease of integration with existing SMPL-X based models we predict the vertices corresponded to hair using distilled version of the ROME with PIXIE [3]. The resulted mesh contains the hair from ROME basis and shoulders from SMPL-X.

We provide an extended cross-driving qualitative comparison with neural-based rendering methods in Figure 4. Then, we provide an additional self-driving qualitative comparison in Figure 5. Most of the results are consistent with the metrics obtained in the main text.

We evaluate the effect that trained offsets have on the quality of rendering. To do that, we remove the head reconstruction step from the training pipeline and only train deferred neural rendering system using base FLAME meshes. The quantitative results are in Table 1. Notice how the quality degrades for the model with no offsets in the hair and shoulders areas and to the overall worse quantitatively measured performance. This leads us to the conclusion that our system for coarse mesh estimation can aid other neural rendering systems produce better quality reconstructions.

1.3 Linear model

We evaluate visual quality in Figure 7. We note that the value of MSE that is achieved by our regressor leads to visually similar reconstructions. Our mesh reconstruction model can achieve up to **10 times** speed-up without any perceived degradation in reconstruction quality.

Additionally, we evaluate the semantic manipulation capabilities of the linear model in a similar way to the face parametric models. Specifically, we pick individual basis vectors and see how varying their coefficient changes the reconstructed mesh. The results can be seen in Figure 6. We observe a certain semantic disentanglement for the first hair and neck basis vectors. This disentanglement allows us to perform mesh and image editing tasks, like the editing of hairstyle, which we show in Figure 8. Here, we obtain this modified reconstruction by simply varying one of the predicted coefficients for the linear basis.

Fig. 2: Extended qualitative comparison on the H3DS dataset. We compare 3D reconstructions and renders obtained using a single **source** image.

Fig. 3: Additional examples of mesh-based avatars creating using a single **source** image, and animated using the camera pose and the expression parameters estimated from the **driver** image. First five rows contains results for samples out of the train distribution.

Fig. 4: Additional comparison of renders on a VoxCeleb2 dataset. The task is to reenact the **source** image with the expression and pose of the **driver** image. This comparison is done in a cross-driving scenario, which we also use for quantitative comparison.

Fig. 5: Additional comparison of renders on a VoxCeleb2 dataset. The task is to reenact the **source** image with the expression and pose of the **driver** image. This comparison is done in a self-driving scenario, which we also use for quantitative comparison.

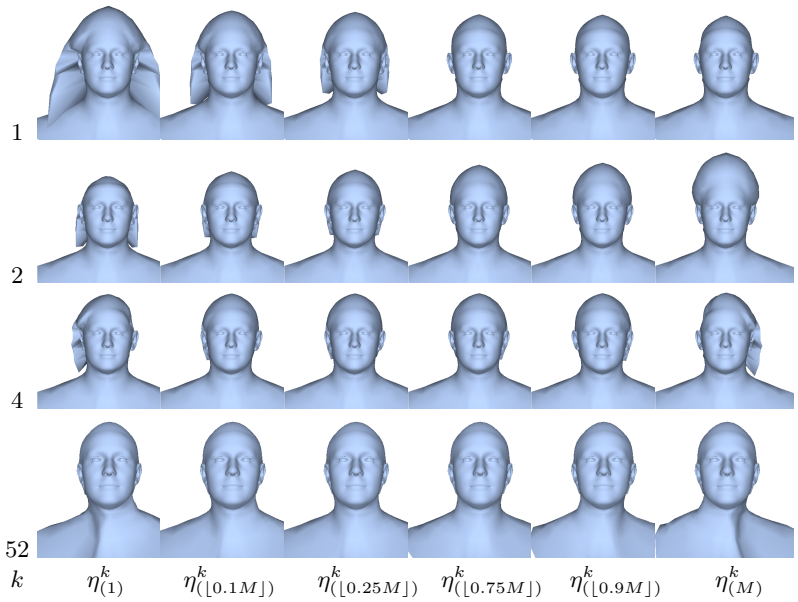


Fig. 6: We show how the estimated meshes can be semantically manipulated by varying the individual components of the PCA basis. We assume that M meshes were initially predicted by the ROME system. Then, we estimate M coefficients $\eta_m \in \mathbb{R}^K$, each is used to reconstruct m-th mesh via the PCA basis. In our experiments, the set of 50 basis vectors are used to reconstruct the hair, and the set of 10 basis vectors are used to reconstruct the neck and the shoulders. We denote each component of η_m as η^k_m. Then, we use the mean vector $\eta = \frac{1}{M}\eta_m$ to reconstruct the base mesh, and modify its k-th component to the p-th order statistic $\eta^k_{(p)}$ over the dataset $\{\eta^k_m\}^M_{m=1}$, where $p \in \{1, \ldots, M\}$. We show the resulting reconstructions above. Each row corresponds to a component which is modified, and each column corresponds to its new value (minimum, 10%, 25%, 75%, 90%, maximum). The first three lines correspond to the hair components, and the last line – to the neck component.

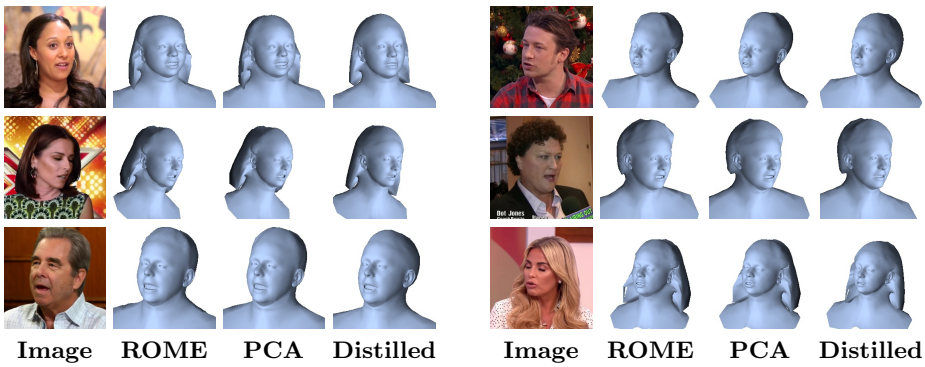


Fig. 7: We provide more examples to qualitatively evaluate the performance of the distilled linear model.

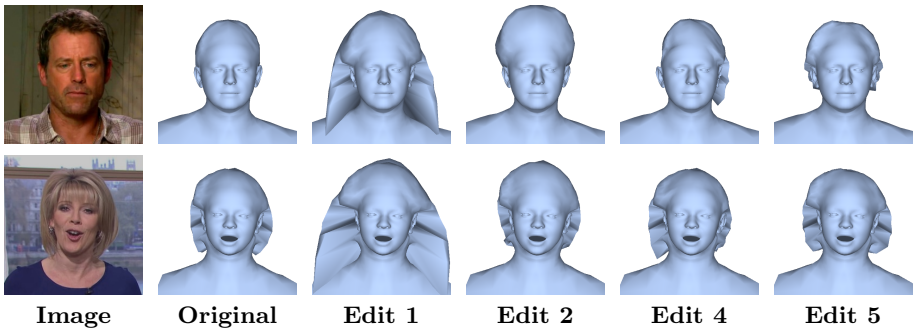


Fig. 8: Examples of hair style manipulation using only individual components of a PCA basis.

References

1. Cao, Q., Shen, L., Xie, W., Parkhi, O.M., Zisserman, A.: Vggface2: A dataset for recognising faces across pose and age. 2018 13th IEEE International Conference on Automatic Face & Gesture Recognition (FG 2018) pp. 67–74 (2018) 1
2. Cortacero, K., Fischer, T., Demiris, Y.: Rt-bene: A dataset and baselines for real-time blink estimation in natural environments. 2019 IEEE/CVF International Conference on Computer Vision Workshop (ICCVW) pp. 1159–1168 (2019) 1
3. Feng, Y., Choutas, V., Bolkart, T., Tzionas, D., Black, M.: Collaborative regression of expressive bodies using moderation. In: International Conference on 3D Vision (3DV) (2021) 4
4. He, K., Zhang, X., Ren, S., Sun, J.: Identity mappings in deep residual networks. ECCV (2016) 2
5. Isola, P., Zhu, J.Y., Zhou, T., Efros, A.A.: Image-to-image translation with conditional adversarial networks. 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR) pp. 5967–5976 (2017) 2
6. Karras, T., Laine, S., Aila, T.: A style-based generator architecture for generative adversarial networks. 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) pp. 4396–4405 (2019) 2
7. Miyato, T., Kataoka, T., Koyama, M., Yoshida, Y.: Spectral normalization for generative adversarial networks. ArXiv **abs/1802.05957** (2018) 2
8. Qiao, S., Wang, H., Liu, C., Shen, W., Yuille, A.L.: Weight standardization. ArXiv **abs/1903.10520** (2019) 2
9. Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M.S., Berg, A.C., Fei-Fei, L.: Imagenet large scale visual recognition challenge. International Journal of Computer Vision (2015) 1
10. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. International Conference on Learning Representations (2015) 1
11. Ulyanov, D., Vedaldi, A., Lempitsky, V.S.: Instance normalization: The missing ingredient for fast stylization. ArXiv **abs/1607.08022** (2016) 2
12. Wang, T.C., Liu, M.Y., Tao, A., Liu, G., Kautz, J., Catanzaro, B.: Few-shot video-to-video synthesis. Advances in Neural Information Processing Systems (NeurIPS) (2019) 2
13. 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. 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition pp. 8798–8807 (2018) 2
14. Wu, Y., He, K.: Group normalization. In: ECCV (2018) 2