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

Abstract. We present a system for realistic one-shot mesh-based human head avatars creation, ROME for short. Using a *single* photograph, our model estimates a person-specific head mesh and the associated neural texture, which encodes both local photometric and geometric details. The resulting avatars are rigged and can be rendered using a neural network, which is trained alongside the mesh and texture estimators on a dataset of in-the-wild videos. In the experiments, we observe that our system performs competitively both in terms of head geometry recovery and the quality of renders, especially for the cross-person reenactment.

1 Introduction

Personalized human avatars are becoming a key technology across several application domains, such as telepresence, virtual worlds, and online commerce. In many practical cases, it is sufficient to personalize only a part of the avatar's body, while the remaining areas can then be picked from a pre-defined library of assets or omitted from the interface. Towards this end, many applications require personalization at the head level, i.e., the creation of person-specific head models, thus making it an important and viable intermediate step between personalizing only the face and the entire body. Alone, face personalization is often insufficient, while the full-body reconstruction remains a complicated task and leads to the reduced quality of the models or requires cumbersome data collection.

Acquiring human avatars from a single photograph (in a "one-shot" setting) offers the highest convenience for the end-user. However, their creation process is particularly challenging and requires strong priors on human geometry and appearance. To this end, parametric models are long known to offer good personalization solutions [1] and were recently adapted to one-shot performance [2,3,4]. Such models can be learned from a relatively small dataset of 3D scans and represent geometry and appearance via textured meshes, making them compatible with many computer graphics applications and pipelines. However, they cannot be trivially expanded to the whole head region due to the large geometric variability of the non-facial parts such as hair and neck.

Our proposed system addresses this issue and allows parametric face models to represent the non-facial parts. In order to handle the increased geometric and photometric variability, we train our method on a large dataset of in-the-wild

Fig. 1: Our system creates realistic mesh-based avatars from a single **source** photo. These avatars are rigged, i.e., they can be driven by the animation parameters from a different **driving** frame. At the same time, our obtained **meshes** and **renderes** achieve a high degree of personalization in both appearance and geometry and are trained in an end-to-end fashion on a dataset of in-the-wild videos without any additional 3D supervision.

videos [5] and use neural networks to parameterize both the geometry and the appearance. For the appearance modeling, we follow the deferred neural rendering [6] paradigm and employ a combination of neural textures and rendering networks. In addition, a neural rendering framework [7] is used to enable end-to-end training and achieve high visual realism of the resulting head models. After training, the geometric and appearance networks can be conditioned on the information extracted from a single photograph, enabling one-shot realistic avatar generation.

To the best of our knowledge, our system is the first that is capable of creating realistic personalized human head models in a rigged mesh format from a single photograph. This distinguishes our model from a growing class of approaches that a) recover neural head avatars without explicit geometry [8,9,10,11], b) can personalize the face region but not the whole head [12,13,1,2], and c) from commercial systems that create non-photorealistic mesh avatars from a single image [14,15]. Alongside our main model, we also discuss its simplified version based on a linear blendshapes basis and show how to train it using the same video dataset. Below, we refer to the avatars generated by our system as ROME avatars (Realistic One-shot Mesh-based avatars).

2 Related work

Parametric models of human faces. Over the recent decades, 3D face reconstruction methods have been actively employed to tackle the problems of face tracking and alignment [16,3], face recognition [17,18], and generative modelling [12,13,19,20,21,22]. In all these scenarios, statistical mesh-based models, aka parametric models [1], remain one of the widely used tools [23,24]. State-of-the-art parametric models for human heads consist of rigged meshes [25] which support a diverse range of animations via disentangled shape and expression

blendshapes and rigid motions for the jaw, neck, and eyeballs. However, they only provide reconstructions for the face, ears, neck, and forehead regions, limiting the range of applications. Including full head reconstruction (i.e., hair and shoulders) into these parametric models is possible, but existing approaches require significantly more training data to be gathered in the form of 3D scans. Instead, in our work, we propose to leverage existing large-scale datasets [5] of in-the-wild videos via the learning-by-synthesis paradigm without any additional 3D annotations.

Neural 3D human head models. While parametric models provide sufficient reconstruction quality for many downstream applications, they are not able to depict the fine appearance details that are needed for photorealistic modelling. In recent years, the problem of representing complex geometry and appearance of humans started being addressed using high-capacity deep neural networks. Some of these works use strong human-specific priors [20,2,26,27], while others fit high-capacity networks to data without the use of such priors [28,29,19,22,21,30,31]. The latter methods additionally differ by the type of data structure used to represent the geometry, namely, mesh-based [2,22,21,32], point-based [28,33], and implicit models [19,29,20,26,31,27,34]. Additionally, recently there have emerged the hybrid models [35,36] where authors integrate face priors from parametric models with implicit representations to learn geometry and rendering for the specific person from the video.

However, mesh-based models arguably represent the most convenient class of methods for downstream applications. They provide better rendering quality and better temporal stability than point-based neural rendering. Also, unlike methods based on implicit geometry, mesh-based methods preserve topology and rigging capabilities and are much faster during fitting and rendering. However, current mesh-based methods either severely limit the range of deformations [2], making it infeasible to learn more complex geometry like hair and clothed shoulders, operate in the multi-shot setting [32] or require 3D scans as training data [22,21]. Our proposed method is also mesh-based, but we allow the prediction of complex deformations without 3D supervision and using a single image, lifting the limitations of the previous and concurrent works.

One-shot neural head models. Advances in neural networks also led to the development of methods that directly predict images using large ConvNets operating in the 2D image domain, with effectively no underlying 3D geometry [9,10,11] or very coarse 3D geometry [8]. These methods achieve state-of-the-art realism [8], use in-the-wild images or videos with no 3D annotations for training, and can create avatars from a single image. However, the lack of an explicit geometric model makes these models incompatible with many real-world applications and limits the span of camera poses that these methods can handle.

Neural mesh rendering. Recently, approaches that combine explicit data structures (point clouds or meshes) with neural image generation have emerged. These methods gained popularity thanks to the effectiveness of the pioneering Deferred Neural Rendering system [6], as well as recent advances in differentiable mesh rendering [7,37,38]. Neural mesh rendering uses 2D convolutional networks

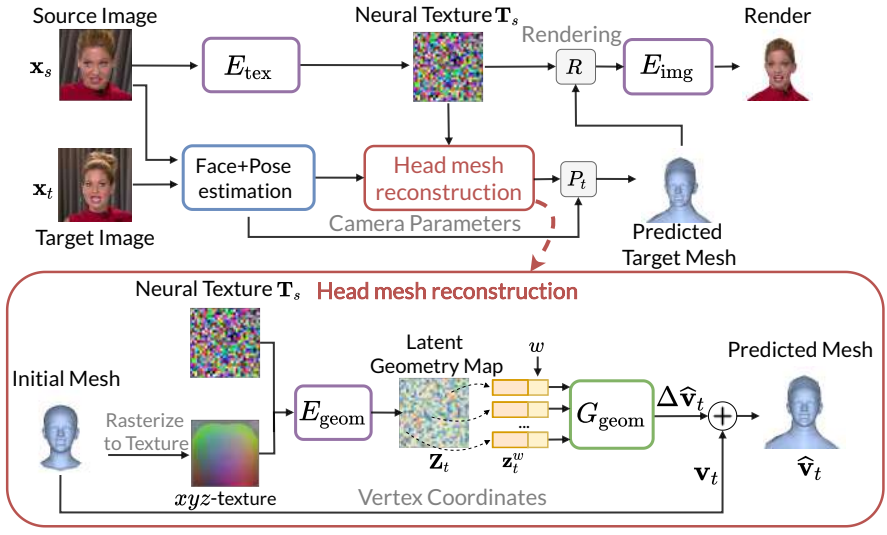


Fig. 2: Overview of our approach and the detailed scheme of the head mesh reconstruction. Given the source photo, we first estimate a *neural texture* that encodes local geometric and photometric details of visible and occluded parts. We then use a pre-trained system [2] for face reconstruction to estimate an initial mesh with a reconstructed facial part. We call this step face and 3D pose estimation. During head mesh reconstruction (bottom), using the estimated neural texture and the initial mesh, we predict the offsets for the mesh vertices, which do not correspond to a face. The offsets are predicted with a combination of a convolutional network E_{geom} and a perceptron network G_{geom}. We then render the personalized head mesh using the camera parameters, estimated by a pre-trained regressor [2] while superimposing the predicted neural texture. Finally, the rendering network E_{img} estimates the RGB image and the mask from the render.

to model complex photometric properties of surfaces. It achieves high realism of renders with fine details present even when they are missing in the underlying geometric model. In this work, we adapt these advances to human head modelling while training using a large dataset of in-the-wild videos.

3 Method

Our goal is to build a system that jointly learns to produce photorealistic renders of human heads and estimate their 3D meshes using only a *single image* without any 3D supervision.

To achieve that, we use a large-scale dataset [5] of in-the-wild videos with talking speakers. All frames in each video are assumed to depict the same person in the same environment (defined by lighting, hairstyle, and person's clothing).

At each training step, we sample two random frames $\mathbf{x}_s$ and $\mathbf{x}_t$ from a random training video. Our goal is to reconstruct and render the target image $\hat{\mathbf{x}}_t$ given a) the personal details and the face shape extracted from the source image $\mathbf{x}_s$, as well as b) the head pose, the facial expression, and the camera pose estimated from the target image $\mathbf{x}_t$. The final reconstruction loss is backpropagated and used to update the parameters of the model's components.

After training, we can create a personalized head model by estimating all parameters from a single image. This model can then be *animated* using face tracking parameters extracted from any talking head sequence and rendered from a range of viewpoints similar to those present in the training dataset (Figure 1).

3.1 Model overview

In our model, we jointly train multiple neural networks that perform rendering and mesh reconstruction. The training pipeline proceeds as follows (Figure 2):

Neural texture estimation. The source image $\mathbf{x}_s$ is encoded into a neural texture $\mathbf{T}_s$, which describes both person-specific appearance and geometry. The encoding is done by a convolutional neural network E_{tex}.

Face and 3D pose estimation. In parallel, we apply a pre-trained DECA system [2] for face reconstruction to both the source and the target image, which estimates facial shape, expression, and head pose. Internally, it uses the FLAME parametric head model [25], which includes mesh topology, texture mapping, and blendshapes. We use the face shape from the source image $\mathbf{x}_s$ as well as the facial expression and the camera pose from the target image $\mathbf{x}_t$ for further processing.

Head mesh reconstruction. The vertices of the DECA mesh with personalized face region and generic non-facial parts are rendered into an xyz-coordinate texture using the predefined texture mapping. The xyz-texture and the neural texture $\mathbf{T}_s$ are concatenated and processed with the U-Net network [39] E_{geom} into a new texture map $\mathbf{Z}_t$, called *latent geometry* map. The 3D displacements for each mesh vertex are then decoded independently by the multi-layer perceptron G_{geom} that predicts a 3D offset $\Delta\hat{\mathbf{v}}$ for each vertex. This step reconstructs the personalized model for non-face parts of the head mesh. The obtained reconstructions are compatible with the topology/connectivity of the FLAME mesh [25].

Deferred neural rendering. The personalized head mesh is rendered using the pose estimated by DECA for the target image and with the superimposed neural texture. The rendered neural texture and the rasterized surface normals are concatenated and processed by the decoding (rendering) U-Net network E_{img} to predict the rendered image $\hat{\mathbf{x}}_t$ and the segmentation mask $\hat{\mathbf{s}}_t$. During training, the difference between the predictions and the ground truths is used to update all components of our system.

Below we discuss our system and its training process in more detail. We also describe a training procedure for a simplified version of our model, which represents head geometry using a linear basis of blendshapes.

3.2 Parametric face modeling

Our method uses a predefined head mesh with the corresponding topology, texture coordinates w, and rigging parameters, which remain fixed for all avatars. More specifically, we use FLAME [25] head model that has N base vertices $\mathbf{v}_{\text{base}} \in \mathbb{R}^{3N}$, and two sets of K and L basis vectors (blendshapes) that encode shape $\mathcal{B} \in \mathbb{R}^{3N \times K}$ and expression $\mathcal{D} \in \mathbb{R}^{3N \times L}$. The reconstruction process is carried out in two stages. First, the basis vectors are blended using the person- and expression-specific vectors of linear coefficients ϕ and ψ. Then, the linear blend skinning [25] function $\mathcal{W}$ is applied, parameterized by the angles θ, which rotates the predefined groups of vertices around linearly estimated joints. The final reconstruction in world coordinates can be expressed as follows:

$$\mathbf{v}(\phi, \psi, \theta) = \mathcal{W}\Big(\mathbf{v}_{\text{base}} + \mathcal{B}\phi + \mathcal{D}\psi,\ \theta\Big).$$

In previous works [12], a similar set of parameters for the 3DMM [1] parametric model was obtained via photometric optimization. More recently, learning-based methods [2,3] capable of feed-forward estimation started to emerge. In our work, given an input image, we use a pre-trained feed-forward DECA system [2] to estimate ϕ, ψ, θ, and the camera parameters.

During training, we apply DECA to both source image $\mathbf{x}_s$ and the target image $\mathbf{x}_t$. The face shape parameters ϕ_s from the source image $\mathbf{x}_s$ alongside the expression ψ_t, head pose θ_t and camera parameters from the target image $\mathbf{x}_t$ are then used to reconstruct the initial FLAME vertices $\mathbf{v}_t = \mathbf{v}(\phi_s, \psi_t, \theta_t)$, as well as camera transform $\mathcal{P}_t$.

3.3 Head mesh reconstruction

The FLAME vertices $\mathbf{v}_t$ estimated by DECA provide good reconstructions for the face region but lack any person-specific details in the remaining parts of the head (hair and shoulders). To alleviate that, we predict person-specific mesh offsets for non-facial regions while preserving the face shape predicted by DECA. We additionally exclude ear regions since their geometry in the initial mesh is too complex to be learned from in-the-wild video datasets.

These mesh offsets are estimated in two steps. First, we encode both the xyz-coordinate texture and the neural texture $\mathbf{T}_s$ into the latent geometry texture map $\mathbf{Z}_t$ via a U-Net network E_{geom}. It allows the produced latent map to contain both positions of the initial vertices $\mathbf{v}_t$ and their semantics, provided by the neural texture.

From $\mathbf{Z}_t$ we obtain the vectors $\mathbf{z}_t^w$ by bilinear interpolation at the fixed texture coordinates w. The vectors $\mathbf{z}_t^w$ and their coordinates w are then concatenated and passed through a multi-layer perceptron G_{geom} to predict the coefficients $\hat{\mathbf{m}}_t \in \mathbb{R}^{3N \times 3}$ independently for each vertex in the mesh. These coefficients are multiplied elementwise by the normals $\mathbf{n}_t$, calculated for each vertex in $\mathbf{v}_t$, to obtain the displacements: $\Delta\hat{\mathbf{v}}_t = \hat{\mathbf{m}} \odot \mathbf{n}_t$. These displacements are then zeroed out for face and ear regions, and the final reconstruction in world coordinates is obtained as follows: $\hat{\mathbf{v}}_t = \mathbf{v}_t + \Delta\hat{\mathbf{v}}_t$.

3.4 Deferred neural rendering

We render the reconstructed head vertices $\hat{\mathbf{v}}_t$ using the topology and texture coordinates w from the FLAME model with the superimposed neural texture $\mathbf{T}_s$. For that, we use a differentiable mesh renderer $\mathcal{R}$ [7] with the camera transform $\mathcal{P}_t$ estimated by DECA for the target image $\mathbf{x}_t$.

The resulting rasterization, which includes both the neural texture and the surface normals, is processed by the rendering network E_{img} to obtain the predicted image $\hat{\mathbf{x}}_t$ and the segmentation mask $\hat{\mathbf{s}}_t$. E_{img} consists of two U-Nets that separately decode an image and a mask. The result of the deferred neural rendering is the reconstruction of the target image $\hat{\mathbf{x}}_t$ and its mask $\hat{\mathbf{s}}_t$, which are compared to the ground-truth image $\mathbf{x}_t$ and mask $\mathbf{s}_t$ respectively.

3.5 Training objectives

In our approach, we learn the geometry of hair and shoulders, which are not reconstructed by the pre-trained DECA estimator, without any ground-truth 3D supervision during training. For that we utilize two types of objectives: segmentation-based geometric losses $\mathcal{L}_{\text{geom}}$ and photometric losses $\mathcal{L}_{\text{photo}}$.

We found that explicitly assigning subsets of mesh vertices to the neck and the hair regions helps a lot with the quality of final deformations. It allows us to introduce a topological prior for the predicted offsets, which is enforced by.

To evaluate the geometric losses, we calculate two separate occupancy masks using a soft rasterization operation [37]. First, $\hat{\mathbf{o}}_t^{\text{hair}}$ is calculated with detached neck vertices, so that the gradient flows through that mask only to the offsets corresponding to the hair vertices, and then $\hat{\mathbf{o}}_t$ is calculated with detached hair vertices. We match the hair occupancy mask to the ground-truth mask $\mathbf{s}_t^{\text{hair}}$ (which covers the hair, face, and ears), and the estimated occupancy mask to the whole segmentation mask $\mathbf{s}_t$: $\mathcal{L}_{\text{occ}} = \lambda_{\text{hair}} \left\| \hat{\mathbf{o}}_t^{\text{hair}} - \mathbf{s}_t^{\text{hair}} \right\|_2^2 + \lambda_{\text{o}} \left\| \hat{\mathbf{o}}_t - \mathbf{s}_t \right\|_2^2$.

We also use an auxiliary Chamfer loss to ensure that the predicted mesh vertices cover the head more uniformly. Specifically, we match the 2D coordinates of the mesh vertices projected into the target image to the head segmentation mask. We denote the subset of predicted mesh vertices, visible in the target image, as $\hat{\mathbf{p}}_t = \mathcal{P}'_t(\hat{\mathbf{v}}_t)$, and the number of these vertices as N_t, so that $\hat{\mathbf{p}}_t \in \mathbf{R}^{N_t \times 2}$. Notice that operator $\mathcal{P}'_t$ here not only does the camera transformation but also discards the z coordinate of the projected mesh vertices. To compute the loss, we then sample N_t 2D points from the segmentation mask $\mathbf{s}_t$ and estimate the Chamfer distance between the sampled set of points $\mathbf{p}_t$ and the vertex projections:

$$\mathcal{L}_{\text{chm}} = \frac{1}{2N_t} \sum_{\hat{p}_t \in \hat{\mathbf{p}}_t} \left\| \hat{p}_t - \arg\min_{p \in \mathbf{p}_t} \left\| p - \hat{p}_t \right\| \right\| + \frac{1}{2N_t} \sum_{p_t \in \mathbf{p}_t} \left\| p_t - \arg\min_{\hat{p} \in \hat{\mathbf{p}}_t} \left\| \hat{p} - p_t \right\| \right\|.$$

Lastly, we regularize the learned geometry using the Laplacian penalty [40]. Initially, we found that regularizing offsets $\Delta\hat{\mathbf{v}}$ worked better than regularizing full coordinates $\hat{\mathbf{v}}$ and stuck to that approach for all experiments. Our version of the Laplacian loss can be written as:

$$\mathcal{L}_{\text{lap}} = \frac{1}{V}\sum_{i=1}^{V}\left\|\Delta\hat{\mathbf{v}}_i - \frac{1}{\mathcal{N}(i)}\sum_{j\in\mathcal{N}(i)}\Delta\hat{\mathbf{v}}_j\right\|_1,$$

where $\mathcal{N}(i)$ denotes a set indices for vertices adjacent to the i-th vertex in the mesh.

We also use photometric optimization that matches the predicted and the ground truth images. This allows us to obtain photorealistic renders and aid in learning proper geometric reconstructions. We utilize perceptual loss $\mathcal{L}_{\text{per}}$ [41], the face recognition loss $\mathcal{L}_{\text{idt}}$ [42] and the multi-resolution adversarial loss $\mathcal{L}_{\text{adv}}$ [43,44]. We also use the Dice loss $\mathcal{L}_{\text{seg}}$ [45] to match the predicted segmentation masks.

The final objective is weighted sum of the geometric and the photometric losses described above.

3.6 Linear deformation model

In addition to the full non-linear model introduced above, we consider a simplified parametric model with a linear basis of offsets $\Delta\hat{\mathbf{v}}$. While this model is similar to parametric models [25,46], we still do not use 3D scans for training and instead obtain our linear model by "distilling" the non-linear model. Additionally, we train a feed-forward estimator that predicts the linear coefficients from the input image.

The motivation for training this additional model is to show that the deformations learned by our method can be approximated using a system with a significantly lower capacity. Such a simple regression model can be easier to apply for inference on low-performance devices.

To train the linear model, we first obtain the basis of offsets $\mathcal{F} \in \mathbb{R}^{3N\times K}$, which is similar to the blendshapes used in the FLAME parametric model. This basis is obtained by applying a low-rank PCA [47] to the matrix of offsets $\Delta\hat{\mathbf{V}} \in \mathbb{R}^{3N\times M}$, calculated using M images from the dataset. Following [25], we discard most of the basis vectors and only keep K components corresponding to maximal singular values. The approximated vertex offsets $\tilde{\mathbf{v}}$ for each image can then be estimated as following $\tilde{\mathbf{v}} = \mathcal{F}\eta$, where η is obtained by applying the pseudo-inverse of a basis matrix $\mathcal{F}$ to the corresponding offsets $\Delta\hat{\mathbf{v}}$: $\eta = \left(\mathcal{F}^T\mathcal{F}\right)^{-1}\mathcal{F}^T\Delta\hat{\mathbf{v}}$

We then train the regression network by estimating a vector of basis coefficients η_t, given an image $\mathbf{x}_t$. For that, we minimize the mean square error (MSE) loss $\|\hat{\eta}_t - \eta_t\|_2^2$ between the estimated coefficients and the ground truth. Also, we set the segmentation loss $\mathcal{L}_{\text{occ}}$ on hair and neck masks, keypoint loss to accelerate learning the pose and camera and Chamfer $\mathcal{L}_{\text{chm}}$ to pull samples from predicted and ground truth meshes.

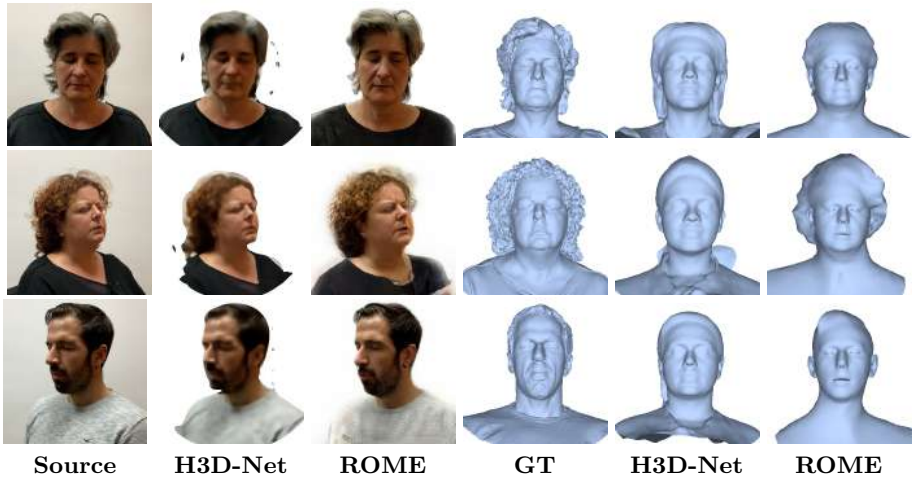


Fig. 3: Qualitative comparison of the representative cases from the H3DS dataset. While neither of the two methods achieves perfect results, arguably, ROME achieves more realistic renders and better matches the head geometry than H3D-Net in the single-shot mode. Furthermore, an important advantage of ROME is that the resulting avatars are ready for animation and are obtained in a feed-forward manner without the lengthy fine-tuning process employed by H3D-Net.

4 Experiments

We train our models on the VoxCeleb2 [5] dataset of videos. This large-scale dataset contains an order of 10^5 videos of 10^3 different speakers. It is widely used [48,8,11] to train talking head models. However, the main drawback of this dataset is the mixed quality of videos and the heavy bias towards frontal poses.

To address these well-known limitations, we process this dataset using an off-the-shelf image quality analysis model [49] and a 3D face-alignment network [50]. We then filter out the data which has poor quality and non-diverse head rotations. Our final training dataset has ≈ 15000 sequences. We note that filtering/pruning does not fully solve the problem of head rotation bias, and our method still works best in frontal views. For more details, please refer to the supplementary materials.

We also use the H3DS [20] dataset of photos with associated 3D scans to evaluate the quality of head reconstructions.

4.1 Implementation details

In the experiments, unless noted otherwise, we train all architectures jointly and end-to-end. We use the following weights: $\lambda_{\text{hair}} = 10$, $\lambda_{\text{per}} = 1$, $\lambda_{\text{idt}} = 0.1$, $\lambda_{\text{adv}} = 0.1$, $\lambda_{\text{seg}} = 10$, and enable the neck and the 2D Chamfer loss $\lambda_{\text{chm}} = 0.01$) and $\lambda_{\text{lap}} = 10$. We ablate all geometry losses and method parts below.

Table 1: Evaluation results on the H3DS dataset in the one-shot scenario for our models, H3D-Net, and DECA. We compute Chamfer distance (lower is better) across all available scans, reconstructed from three different viewpoints. Both of the ROME variants significantly exceed H3D-Net in the one-shot reconstruction quality.

Method	DECA	H3D-Net	ROME	LinearROME
Chamfer Distance	15.0	15.1	12.6	12.5

We train our models at 256×256 resolution using ADAM [51] with the fixed learning rate of 10^{-4}, $\beta_1 = 0$, $\beta_2 = 0.999$, and a batch size of 32. For more details, please refer to the supplementary materials.

4.2 Evaluation

3D reconstruction. We evaluate our head reconstruction quality using a novel H3DS dataset [20]. We compare against the state-of-the-art head reconstruction method H3D-Net [20], which uses signed distance functions to represent the geometry. While providing great reconstruction quality in the sparse-view scenario, their approach has several limitations. For example, H3D-Net requires a dataset of full head scans to learn the prior on head shapes. Additionally, its results do not have fixed topology or rigging and their method requires fine-tuning per scene, while our method works in a feed-forward way.

We carry out the comparison with H3D-Net in a single-view scenario, which is native for our method but is beyond the capabilities stated by the authors in the original publication [20]. However, to the best of our knowledge, H3D-Net is the closest system to ours in single-view reconstruction capabilities (out of all systems with either their code or results available). Additionaly, we tried to compare our system with PIFuHD [52], which unfortunately failed to work with heads images without body (see supplementary).

We show qualitative comparison in Figure 3. We evaluate our method and H3D-Net both for frontal- and side-view reconstruction. We note the significant overfitting of H3D-Net to the visible hair geometry, while our model provides reconstructions more robust to the change of viewpoint.

In total, we compared our models on all scans available in the test set of the H3DS dataset, and each scan was reconstructed from three different viewpoints. We provide the measured mean Chamfer distance both for our method and baselines across all scans in Tab. 1.

Rendering. We evaluate the quality of our renders on the hold-out subset VoxCeleb2 dataset. We use a cross-driving comparison scenario for qualitative comparison to highlight the animation capabilities of our method, and self-driving scenario for quantitative comparison.

First, we compare with a FLAMETex [25] rendering system, which works explicitly with mesh rendering. From the source image, FLAMETex estimates the albedo via a basis of RGB textures, and then combines it with predicted

Fig. 4: Comparison of renders on a VoxCeleb2 dataset. The task is to reenact the **source** image with the expression and pose of the **driver** image. Here, we picked diverse examples in terms of pose variation to highlight the differences in performance of compared methods. We observe that for the large head pose rotations, purely neural-based methods (**FOMM**, **Bi-Layer**) struggle to maintain consistent quality. In contrast, our rendering method produces images that are more robust to pose changes. Admittedly, for small pose changes, neural-based methods exhibit a smaller identity gap than ROME (bottom row) and overall outperform our method in terms of rendering quality. As a reference, we also include a non-neural **FLAMETex** rendering method, which is employed in state-of-the-art one-shot face reconstruction systems [2] but is not able to personalize the avatar at the head level.

scene-specific shading. In contrast, our method predicts a rendered image directly and avoids the complexity of explicit albedo-shading decomposition.

We then compare with publicly available geometry-free rendering methods, which were trained on the same dataset. For that, we use the First-Order Motion Model (FOMM) [9], the Bi-Layer Avatar Model [11] and recently proposed Thin-Plate-Spline-Motion-Mode (TPSMM) [53]. Both these systems bypass explicit 3D geometry estimation and rely only on learning the scene structure via the parameters of generative ConvNets. Other methods [8,48], which internally utilize some 3D structures, like camera rotations, were out of the scope of our comparison due to the unavailability of pre-trained models.

We present the qualitative comparison in Figure 4, and a quantitative comparison across a randomly sampled hold-out VoxCeleb2 subset in Table 2. We restrict the comparison to the face and hair region as the shoulder pose is not controlled by our method (driven by DECA parameters), which is admittedly a limitation of our system. We thus mask the results according to the face and hair mask estimated from the ground truth image.

Table 2: Here we present the quantitative results on the VoxCeleb2-HQ dataset in the self-reenactment and cross-reenactment modes. Our ROME system performs on par with FOMM and TPSMM in self-reenactment, notably outperforming them in the most perceptually-plausible LPIPS metrics. On the contrary, in the cross-driving scenario, when the task is complex for pure neural-based systems, our method obtains better results.

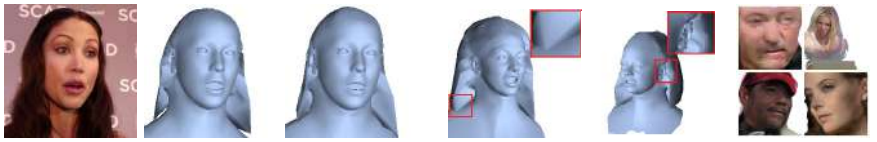
	self-reenactment			cross-reenactment		
Method	LPIPS↓	SSIM↑	PSNR↑	FID↓	CSIM↑	IQA↑
FOMM	0.09	0.87	25.8	52.95	0.53	55.9
Bi-Layer	0.08	0.83	23.7	51.4	0.56	50.48
TPSMM	0.09	0.85	26.1	49.27	0.57	59.5
ROME	0.08	0.86	26.2	45.32	0.62	66.3

Generally, we observe that over the entire test set, the quality of ROME avatars in the self-reenactment mode is similar to FOMM and better than the Bi-layer model. For the cross-reenactment scenario, our model is clearly better than both alternatives according to three metrics, that help to asses unsupervised quality of the images in three aspects: realism, identity preservation and blind quality of the image. The huge gap for IQA [49] and FID [54] is also noticeable in the qualitative comparison, especially for strong pose change (see CSIM [10] column in Tab. 2). The PSNR and SSIM metrics penalize slight misalignments between the sharp ground truth and our renderings much stronger than the blurriness in FOMM reconstructions. The advatage of ROME avatar is noticable even for self-driving case according to LPIPS. We provide a more extensive qualitative evaluation in the supplementary materials.

4.3 Linear basis experiments

As discussed above, we distill our ROME head reconstruction model into a linear parametric model. To do that, we set the number of basis vectors to 50 for the hair and 10 for the neck offsets and run low-rank Principle Component Analysis (PCA) to estimate them. The number of components is chosen to obtain a low enough approximation error. Interestingly, the offsets learned by our model can be compressed by almost two orders of magnitude in terms of degrees of freedom without any practical loss in quality (Figure 5a), which suggests that the capacity of the offset generator is underused in our model. We combine estimated basis vectors with the original basis of the FLAME.

After that, we train feed-forward encoders that directly predict the coefficients of the two basis from the source image. The prediction is performed in two stages. First, face expression, pose and camera parameters are predicted with a MobileNetV2 [55] encoder. Then a slower ResNet-50 encoder [56] is used to predict hair, neck and shape coefficients. The choice of architectures are motivated by the fact that in many practical scenarios only the first encoder needs



(a) Linear model (b) Limitations

Fig. 5: Linear model results and the examples of limitations. On the left, we show how reconstructions learned by our method, **ROME**, could be **distilled** using a linear parametric model. We are able to compress the learned offsets into a small basis, reducing the degrees of freedom by two orders of magnitude. We can then **distill** these offsets using a much faster regression network with a small gap in terms of the reconstruction quality. On the right, we highlight the main limitations of our method, which include the failure related to **long hair** modelling, caused by an incorrect topological prior, no **coverage of ears** and **unrealistic renders** under a significant change of scales.

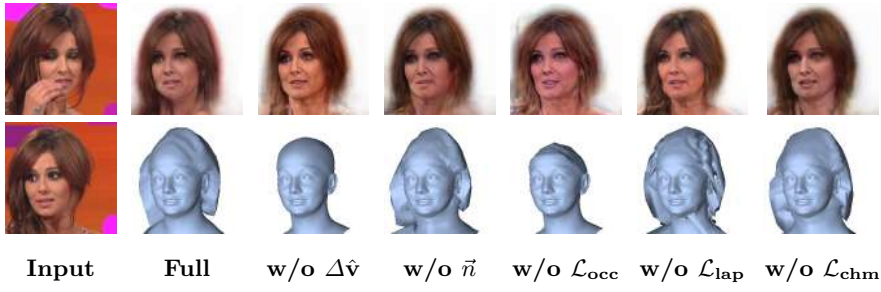


Fig. 6: Ablation study. We qualitatively evaluate the individual components of our *full* model. w/o $\Delta\hat{\mathbf{v}}$: without the per-vertex displacements, we obtain a significantly worse render quality. w/o $\vec{n}$: when we apply per-vertex deformations instead of per-vertex displacements (i.e., deformations alongside the normals), we obtain noisy reconstructions in neck area and worse renders. w/o $\mathcal{L}_{\text{occ}}$: without silhouette-based losses, our model fails to learn proper reconstructions. w/o $\mathcal{L}_{\text{lap}}$: Laplacian regularization smooths the reconstructions. w/o $\mathcal{L}_{\text{chm}}$: chamfer loss allows us to constrain the displaced vertices to lie inside the scene boundaries, which positively affects the smoothness of the visible part of the reconstruction.

to be invoked frequently (per-frame), while the second can run at much lower rate or even only at the model creation time.

4.4 Ablation study

We demonstrate results of ablation study at Figure 6. As expected, predicting more accurate geometry affect the renders (first row). Also, we verify the necessity of all terms of geometry loss. We observe significant improvement in

quality of renders with additional geometry (see Supp 3), which leads us to an optimistic conclusion that our learned coarse mesh may be integrated into other neural rendering systems [8] to improve their quality. Additionally, we observe that geometry losses allows to correctly model coarse details on the hair without noise and reconstruct the hair without sticking with neck. Similar artifacts are removed by adding shifts along the normals.

4.5 Limitations

In some cases, the proposed system often produces somewhat oversmoothed geometry without person-specific attributes. Most of those attributes simply cannot be presented with the limited set of vertices. Our preliminary experiments with subdivision during training did not help to alleviate this problem. It may be interesting to study this problem in a few-shot scenario (given different views) or to use images and proxy geometry in a higher resolution. Another frequent geometry artifact is the roughness of the clothing approximation. As we know, modeling any clothes without 3D data is an extremely difficult task and the fact that our models is not able to do so from videos is not surprising. A principled solution to this problem will likely require learning from datasets that cover these areas (e.g., circular 3D scans of human heads) [57].

Our current model is trained at roughly fixed scale, though explicit geometry modeling allows it to generalize to adjacent scale reasonably well. Still, strong changes of scale lead to poor performance (Figure 5b). More examples are provided in the supplementary materials. Addressing this issue via mip-mapping and multi-scale GAN training techniques remains future work.

Lastly, our model can have artifacts with long hair (Figure 5b, left) or ears (Figure 5b, middle). Handling such cases gracefully are likely to require a departure from the predefined FLAME mesh connectivity to new person-specific mesh topology. Handling such issues using a limited set of pre-designed hair meshes is an interesting direction for future research.

5 Summary

We have presented ROME avatars: a system for creating realistic one-shot mesh-based human head models that can be animated and are compatible with FLAME head models. We compare our model with representative state-of-the-art models from different classes, and show that it is highly competitive both in terms of geometry estimation and the quality of rendering.

Crucially, our system can learn to model head geometry without direct supervision in the form of 3D scans. Despite that, we have observed it to achieve state-of-the-art results in head geometry recovery from a single photograph. At the same time, it also performs better than previous one-shot neural rendering approaches in the cross- and self-driving scenario. We have thus verified that the resulting geometry could be used to improve the rendering quality.

As neural rendering becomes more widespread within graphics systems, ROME avatars and similar systems can become directly applicable, while their one-shot capability and the simplicity of rigging derived from DECA and FLAME could become especially important in practical applications.

Acknowledgements

We sincerely thank Eduard Ramon for providing us the one-shot H3D-Net reconstructions. We also thank Arsenii Ashukha for their comments and suggestions regarding the text contents and clarity, as well as Julia Churkina for helping us with proof-reading. The computational resources for this work were mainly provided by Samsung ML Platform.

References

1. Volker Blanz and Thomas Vetter. A morphable model for the synthesis of 3d faces. In *SIGGRAPH '99*, 1999. 1, 2, 6
2. Yao Feng, Haiwen Feng, Michael J. Black, and Timo Bolkart. Learning an animatable detailed 3d face model from in-the-wild images. *ACM Transactions on Graphics (TOG)*, 40:1 – 13, 2020. 1, 2, 3, 4, 5, 6, 11
3. Jianzhu Guo, Xiangyu Zhu, Yang Yang, Fan Yang, Zhen Lei, and Stan Z Li. Towards fast, accurate and stable 3d dense face alignment. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2020. 1, 2, 6
4. Soubhik Sanyal, Timo Bolkart, Haiwen Feng, and Michael J. Black. Learning to regress 3d face shape and expression from an image without 3d supervision. *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 7755–7764, 2019. 1
5. Joon Son Chung, Arsha Nagrani, and Andrew Zisserman. Voxceleb2: Deep speaker recognition. In *INTERSPEECH*, 2018. 2, 3, 4, 9
6. Justus Thies, Michael Zollhöfer, and Matthias Nießner. Deferred neural rendering: Image synthesis using neural textures. *arXiv: Computer Vision and Pattern Recognition*, 2019. 2, 3
7. Nikhila Ravi, Jeremy Reizenstein, David Novotny, Taylor Gordon, Wan-Yen Lo, Justin Johnson, and Georgia Gkioxari. Accelerating 3d deep learning with pytorch3d. *arXiv:2007.08501*, 2020. 2, 3, 7
8. Ting-Chun Wang, Arun Mallya, and Ming-Yu Liu. One-shot free-view neural talking-head synthesis for video conferencing. *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 10034–10044, 2021. 2, 3, 9, 11, 14
9. Aliaksandr Siarohin, Stéphane Lathuilière, S. Tulyakov, Elisa Ricci, and N. Sebe. First order motion model for image animation. *Advances in Neural Information Processing Systems (NeurIPS)*, 2019. 2, 3, 11
10. Egor Zakharov, Aliaksandra Shysheya, Egor Burkov, and Victor S. Lempitsky. Few-shot adversarial learning of realistic neural talking head models. In *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, 2019. 2, 3, 12
11. Egor Zakharov, Aleksei Ivakhnenko, Aliaksandra Shysheya, and Victor S. Lempitsky. Fast bi-layer neural synthesis of one-shot realistic head avatars. In *ECCV*, 2020. 2, 3, 9, 11
12. J. Thies, M. Zollhöfer, M. Stamminger, C. Theobalt, and M. Nießner. Face2face: Real-time face capture and reenactment of rgb videos. In *Proc. Computer Vision and Pattern Recognition (CVPR), IEEE*, 2016. 2, 6
13. Hyeongwoo Kim, Pablo Garrido, Ayush Tewari, Weipeng Xu, Justus Thies, Matthias Nießner, Patrick Pérez, Christian Richardt, Michael Zollöfer, and Christian Theobalt. Deep video portraits. *ACM Transactions on Graphics (TOG)*, 37(4):163, 2018. 2

14. AvatarSDK. `https://avatarsdk.com/`. 2
15. Pinscreen. `https://www.pinscreen.com/`. 2
16. Tal Hassner, Shai Harel, Eran Paz, and Roee Enbar. Effective face frontalization in unconstrained images. *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4295–4304, 2015. 2
17. Volker Blanz, Sami Romdhani, and Thomas Vetter. Face identification across different poses and illuminations with a 3d morphable model. *Proceedings of Fifth IEEE International Conference on Automatic Face Gesture Recognition*, pages 202–207, 2002. 2
18. A. Tran, Tal Hassner, Iacopo Masi, and Gérard G. Medioni. Regressing robust and discriminative 3d morphable models with a very deep neural network. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1493–1502, 2017. 2
19. Ben Mildenhall, Pratul P. Srinivasan, Matthew Tancik, Jonathan T. Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. In *ECCV*, 2020. 2, 3
20. Eduard Ramon, Gil Triginer, Janna Escur, Albert Pumarola, Jaime Garcia Giraldez, Xavier Giró i Nieto, and Francesc Moreno-Noguer. H3d-net: Few-shot high-fidelity 3d head reconstruction. *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021. 2, 3, 9, 10
21. Stephen Lombardi, Jason M. Saragih, Tomas Simon, and Yaser Sheikh. Deep appearance models for face rendering. *ACM Transactions on Graphics (TOG)*, 37:1 – 13, 2018. 2, 3
22. Stephen Lombardi, Tomas Simon, Jason M. Saragih, Gabriel Schwartz, Andreas M. Lehrmann, and Yaser Sheikh. Neural volumes. *ACM Transactions on Graphics (TOG)*, 38:1 – 14, 2019. 2, 3
23. Bernhard Egger, W. Smith, Ayush Tewari, Stefanie Wuhrer, Michael Zollhöfer, Thabo Beeler, Florian Bernard, Timo Bolkart, Adam Kortylewski, Sami Romdhani, Christian Theobalt, Volker Blanz, and Thomas Vetter. 3d morphable face models—past, present, and future. *ACM Transactions on Graphics (TOG)*, 39:1 – 38, 2020. 2
24. Stylianos Ploumpis, Evangelos Ververas, Eimear O' Sullivan, Stylianos Moschoglou, Haoyang Wang, Nick E. Pears, W. Smith, Baris Gecer, and Stefanos Zafeiriou. Towards a complete 3d morphable model of the human head. In *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2021. 2
25. 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 (TOG)*, 36:1 – 17, 2017. 2, 5, 6, 8, 10
26. Shunsuke Saito, Tomas Simon, Jason M. Saragih, and Hanbyul Joo. Pifuhd: Multi-level pixel-aligned implicit function for high-resolution 3d human digitization. *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 81–90, 2020. 3
27. Stephen Lombardi, Tomas Simon, Gabriel Schwartz, Michael Zollhoefer, Yaser Sheikh, and Jason M. Saragih. Mixture of volumetric primitives for efficient neural rendering. *ACM Transactions on Graphics (TOG)*, 40:1 – 13, 2021. 3
28. Qianli Ma, Shunsuke Saito, Jinlong Yang, Siyu Tang, and Michael J. Black. Scale: Modeling clothed humans with a surface codec of articulated local elements. In *CVPR*, 2021. 3
29. Jeong Joon Park, Peter R. Florence, Julian Straub, Richard A. Newcombe, and S. Lovegrove. Deepsdf: Learning continuous signed distance functions for shape

representation. *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 165–174, 2019. 3

30. Petr Kellnhofer, Lars Jebe, Andrew Jones, Ryan P. Spicer, Kari Pulli, and Gordon Wetzstein. Neural lumigraph rendering. *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021. 3
31. Michael Oechsle, Songyou Peng, and Andreas Geiger. Unisurf: Unifying neural implicit surfaces and radiance fields for multi-view reconstruction. *International Conference on Computer Vision (ICCV)*, 2021. 3
32. Philip-William Grassal, Malte Prinzler, Titus Leistner, Carsten Rother, Matthias Nießner, and Justus Thies. Neural head avatars from monocular rgb videos. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022. 3
33. Ilya Zakharkin, Kirill Mazur, Artur Grigoriev, and Victor S. Lempitsky. Point-based modeling of human clothing. *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, 2021. 3
34. Tarun Yenamandra, Ayush Tewari, Florian Bernard, Hans-Peter Seidel, Mohamed A. Elgharib, Daniel Cremers, and Christian Theobalt. i3dmm: Deep implicit 3d morphable model of human heads. *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 12798–12808, 2021. 3
35. 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 *2022 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022. 3
36. Guy Gafni, Justus Thies, Michael Zollhöfer, and Matthias Nießner. Dynamic neural radiance fields for monocular 4d facial avatar reconstruction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021. 3
37. Shichen Liu, Tianye Li, Weikai Chen, and Hao Li. Soft rasterizer: A differentiable renderer for image-based 3d reasoning. *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 7707–7716, 2019. 3, 7
38. Samuli Laine, Janne Hellsten, Tero Karras, Yeongho Seol, Jaakko Lehtinen, and Timo Aila. Modular primitives for high-performance differentiable rendering. *ACM Transactions on Graphics*, 39(6), 2020. 3
39. Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *MICCAI*, 2015. 5
40. Olga Sorkine-Hornung. Laplacian mesh processing. In *Eurographics*, 2005. 8
41. Justin Johnson, Alexandre Alahi, and Li Fei-Fei. Perceptual losses for real-time style transfer and super-resolution. In *ECCV*, 2016. 8
42. Qiong Cao, Li Shen, Weidi Xie, Omkar M. Parkhi, and Andrew Zisserman. Vggface2: A dataset for recognising faces across pose and age. *2018 13th IEEE International Conference on Automatic Face & Gesture Recognition (FG 2018)*, pages 67–74, 2018. 8, 21
43. Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron C. Courville, and Yoshua Bengio. Generative adversarial nets. In *NIPS*, 2014. 8
44. Ting-Chun Wang, Ming-Yu Liu, Jun-Yan Zhu, Andrew Tao, Jan Kautz, and Bryan Catanzaro. High-resolution image synthesis and semantic manipulation with conditional gans. *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8798–8807, 2018. 8, 21

45. Fausto Milletari, Nassir Navab, and Seyed-Ahmad Ahmadi. V-net: Fully convolutional neural networks for volumetric medical image segmentation. *2016 Fourth International Conference on 3D Vision (3DV)*, pages 565–571, 2016. 8
46. Silvia Zuffi, Angjoo Kanazawa, Tanya Berger-Wolf, and Michael J. Black. Three-d safari: Learning to estimate zebra pose, shape, and texture from images "in the wild", 2019. 8
47. Nathan Halko, Per-Gunnar Martinsson, and Joel A. Tropp. Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions. *SIAM Rev.*, 53:217–288, 2011. 8
48. Michail Christos Doukas, Stefanos Zafeiriou, and Viktoriia Sharmanska. Headgan: Video-and-audio-driven talking head synthesis. 2021. 9, 11
49. Shaolin Su, Qingsen Yan, Yu Zhu, Cui cui Zhang, Xin Ge, Jinqiu Sun, and Yanning Zhang. Blindly assess image quality in the wild guided by a self-adaptive hyper network. *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020. 9, 12
50. Adrian Bulat and Georgios Tzimiropoulos. How far are we from solving the 2d & 3d face alignment problem? (and a dataset of 230,000 3d facial landmarks). In *International Conference on Computer Vision*, 2017. 9
51. Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *International Conference for Learning Representations*, 2015. 10
52. Shunsuke Saito, Tomas Simon, Jason Saragih, and Hanbyul Joo. Pifuhd: Multi-level pixel-aligned implicit function for high-resolution 3d human digitization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, June 2020. 10
53. Jian Zhao and Hui Zhang. Thin-plate spline motion model for image animation. *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022. 11
54. Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. In *Advances in Neural Information Processing Systems*, 2017. 12
55. Mark Sandler, Andrew G. Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2018. 12
56. Kaiming He, X. Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016. 12
57. Thiemo Alldieck, Marcus Magnor, Bharat Lal Bhatnagar, Christian Theobalt, and Gerard Pons-Moll. Learning to reconstruct people in clothing from a single rgb camera. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019. 14
58. Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *International Conference on Learning Representations*, 2015. 21
59. Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael S. Bernstein, Alexander C. Berg, and Li Fei-Fei. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, 2015. 21
60. Kevin Cortacero, Tobias Fischer, and Y. Demiris. Rt-bene: A dataset and baselines for real-time blink estimation in natural environments. *2019 IEEE/CVF Inter-*

national Conference on Computer Vision Workshop (ICCVW), pages 1159–1168, 2019. 21
61. Takeru Miyato, Toshiki Kataoka, Masanori Koyama, and Yuichi Yoshida. Spectral normalization for generative adversarial networks. *ArXiv*, abs/1802.05957, 2018. 21
62. Ting-Chun Wang, Ming-Yu Liu, Andrew Tao, Guilin Liu, Jan Kautz, and Bryan Catanzaro. Few-shot video-to-video synthesis. *Advances in Neural Information Processing Systems (NeurIPS)*, 2019. 21
63. Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A. Efros. Image-to-image translation with conditional adversarial networks. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5967–5976, 2017. 21
64. Yuxin Wu and Kaiming He. Group normalization. In *ECCV*, 2018. 22
65. Siyuan Qiao, Huiyu Wang, Chenxi Liu, Wei Shen, and Alan Loddon Yuille. Weight standardization. *ArXiv*, abs/1903.10520, 2019. 22
66. Dmitry Ulyanov, Andrea Vedaldi, and Victor S. Lempitsky. Instance normalization: The missing ingredient for fast stylization. *ArXiv*, abs/1607.08022, 2016. 22
67. Kaiming He, X. Zhang, Shaoqing Ren, and Jian Sun. Identity mappings in deep residual networks. *ECCV*, 2016. 22
68. Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks. *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4396–4405, 2019. 22
69. Yao Feng, Vasileios Choutas, Timo Bolkart, Dimitrios Tzionas, and Michael Black. Collaborative regression of expressive bodies using moderation. In *International Conference on 3D Vision (3DV)*, 2021. 23

A Supplementary material

A.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 [58] network pre-trained on ImageNet [59] to match the general content and features from a VGG16-based gaze detection network [60] 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 [60].

The face identity loss $\mathcal{L}_{\text{idt}}$ is the cosine distance between the embeddings of a pre-trained face recognition network [42], 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 [44]. We additionally apply spectral normalization [61] to the discriminator network. The configuration of the adversarial training closely follows the related works on image-to-image translation [62]. In particular, we use the PatchGAN [63] 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 [64] paired with weight standardization [65] in all networks to facilitate training with smaller batch sizes. Empirically we found this combination to perform better than standard instance normalization [66].

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 [67]. 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 [68]. 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.

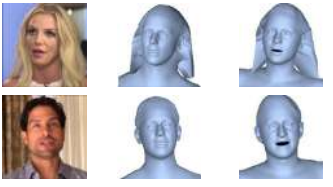
Training details. We use the ADAM optimizer to train all networks in our model. We set the learning rate to $1 \cdot 10^{-4}$ for all the networks except for the discriminator. For it, we set the learning rate to $4 \cdot 10^{-4}$. We also set $\beta_1 = 0$, and $\beta_2 = 0.999$. We train using eight 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.

A.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 8, we present more self-driving results on the H3DS dataset, as well as a side-by-side comparison with H3D-Net. In Figure 9, 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. 7a. 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.

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 [69]. 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 10. Then, we provide an additional self-driving qualitative comparison in Figure 11. 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 3. Notice how the quality degrades for

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

Table 3: 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.

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.

A.3 Linear model

We evaluate visual quality in Figure 13. 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 12. 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 14. Here, we obtain this modified reconstruction by simply varying one of the predicted coefficients for the linear basis.

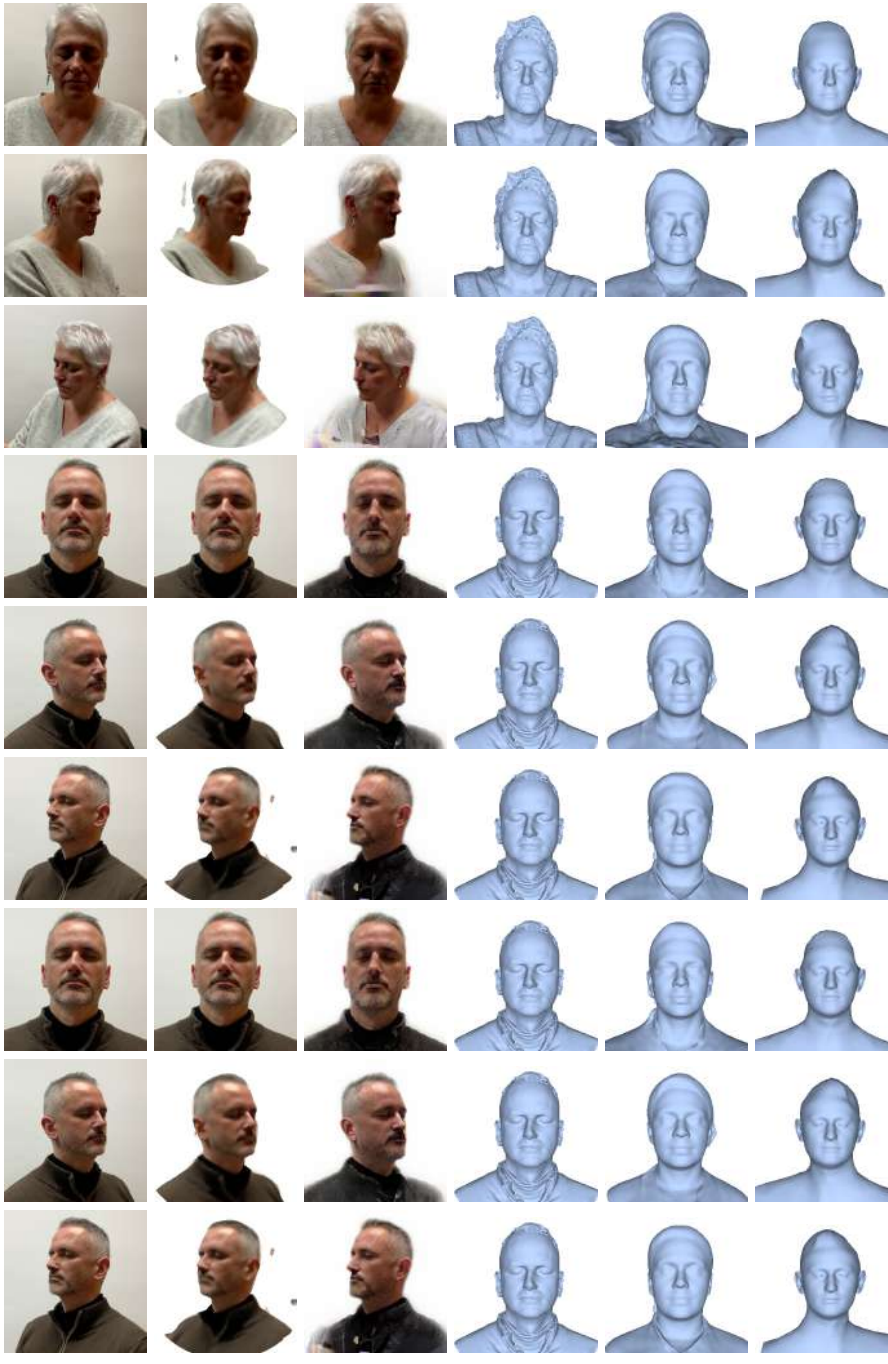


Fig. 8: Extended qualitative comparison on the H3DS dataset. We compare 3D reconstructions and renders obtained using a single **source** image.

Fig. 9: 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.

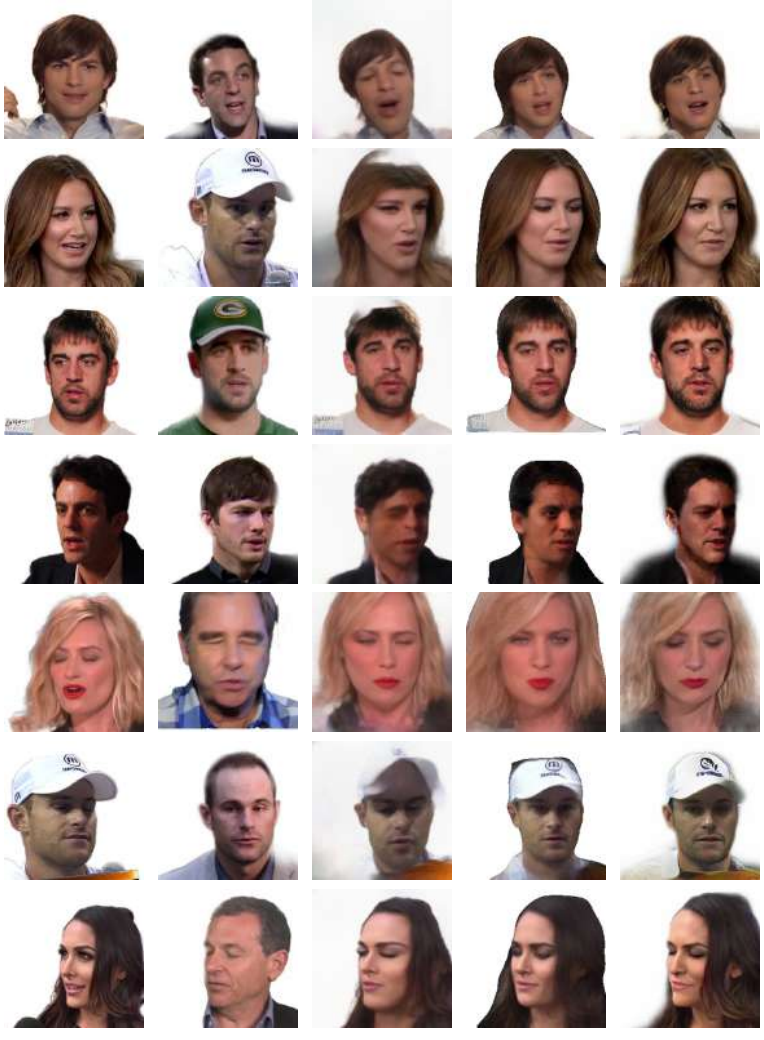


Fig. 10: Additional comparison of renders on a VoxCeleb2 dataset. The task is to reenact the **source** image with the expression and pose of the **driver** image. This comparison is done in a cross-driving scenario, which we also use for quantitative comparison.

Fig. 11: 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.

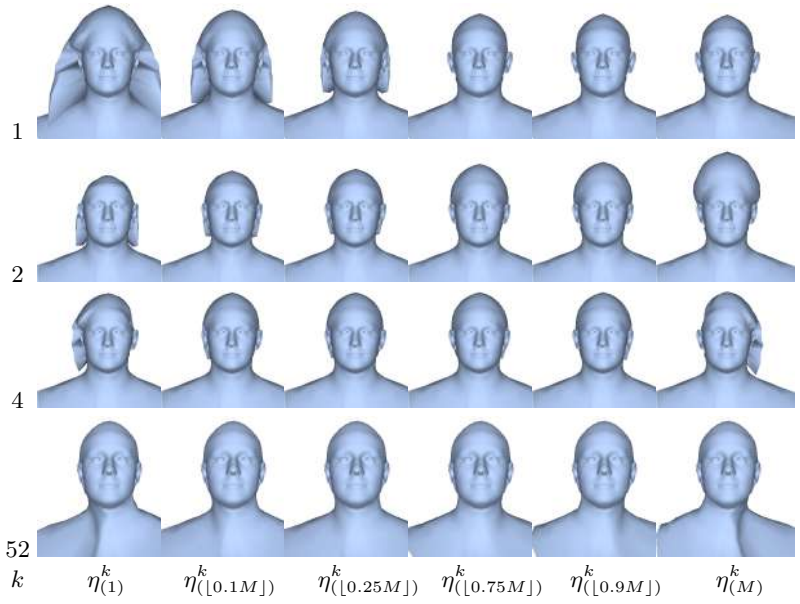


Fig. 12: We show how the estimated meshes can be semantically manipulated by varying the individual components of the PCA basis. We assume that M meshes were initially predicted by the ROME system. Then, we estimate M coefficients $\eta_m \in \mathbb{R}^K$, each is used to reconstruct m-th mesh via the 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. 13: We provide more examples to qualitatively evaluate the performance of the distilled linear model.

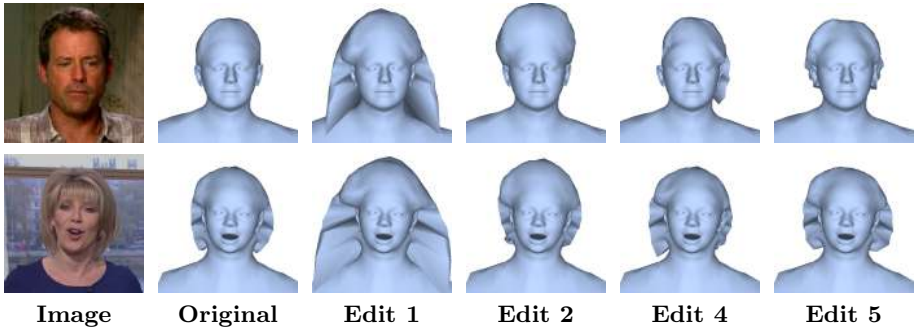


Fig. 14: Examples of hair style manipulation using only individual components of a PCA basis.