HeadCraft: Modeling High-Detail Shape Variations for Animated 3DMMs

Artem Sevastopolsky[1] Philip-William Grassal[2] Simon Giebenhain[1]
ShahRukh Athar[3] Luisa Verdoliva[4,1] Matthias Nießner[1]

[1] Technical University of Munich (TUM), Germany [2] Copresence AG, Germany
[3] Stony Brook University, US [4] University of Naples Federico II, Italy

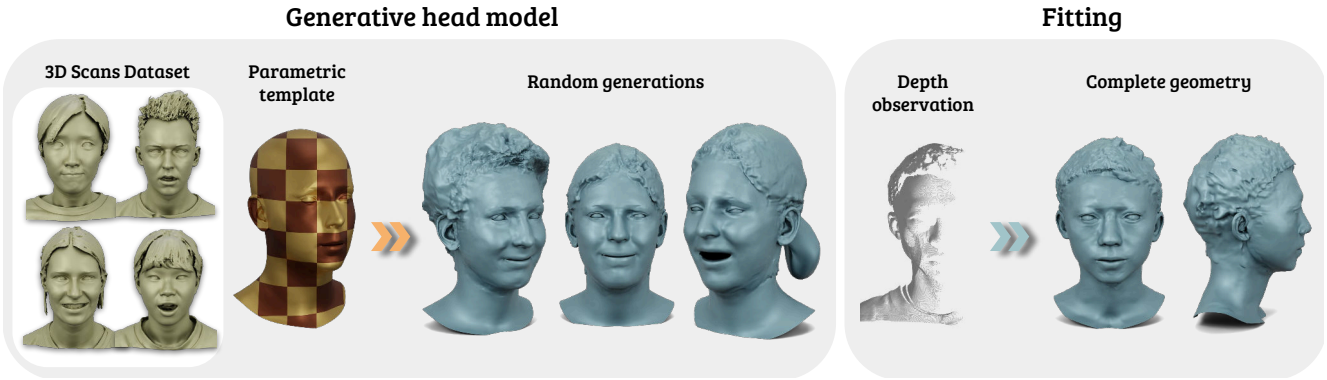

Figure 1. We present HeadCraft[1], a generative model for highly-detailed human heads, ready for animation. Our method is trained on 2D displacement maps collected by registering a parametric template head with free surface displacements to a large set of 3D head scans. The resulting model is highly versatile, which can be demonstrated by fitting the latent code of the model to an arbitrary depth observation.

Abstract

Current advances in human head modeling allow to generate plausible-looking 3D head models via neural representations. Nevertheless, constructing complete high-fidelity head models with explicitly controlled animation remains an issue. Furthermore, completing the head geometry based on a partial observation, e.g. coming from a depth sensor, while preserving details is often problematic for the existing methods. We introduce a generative model for detailed 3D head meshes on top of an articulated 3DMM which allows explicit animation and high-detail preservation at the same time. Our method is trained in two stages. First, we register a parametric head model with vertex displacements to each mesh of the recently introduced NPHM dataset of accurate 3D head scans. The estimated displacements are baked into a hand-crafted UV layout. Second, we train a StyleGAN model in order to generalize over the UV maps of displacements. The decomposition of the parametric model and high-quality vertex displacements allows us to animate the model and modify it semantically. We demonstrate the results of unconditional generation and fitting to the full or partial observation.

1. Introduction

The advent of neural representations, such as NeRFs (Neural Radiance Fields) [38] and SDFs (Signed Distance Functions) [40], has revolutionized the field of 3D modeling by enabling the generation of remarkably realistic 3D head models. These models have found their applications in various domains, including computer graphics, virtual reality, and digital entertainment. The ability to create lifelike 3D head models is crucial for many applications, ranging from video game character design to virtual try-on experiences and medical simulations. While contemporary techniques have made significant strides in achieving high-quality 3D head models, a series of challenges persist, impeding the seamless integration of these models into real-world applications. In particular, one of the primary challenges in the realm of 3D head modeling lies in constructing a neural representation amenable to animation and tracking, while preserving enough detail.

In particular, recently introduced implicit generative models for 3D, such as pi-GAN [14], EG3D [15], or

[1]Project page is available at https://seva100.github.io/headcraft.

StyleSDF [39], fit the distribution of human faces with extraordinary detail, and e.g. in [7, 16], the modeled region is extended to the whole head. To introduce the animation capabilities, these methods require ad-hoc modifications such as introducing deformations to canonical space [8] or driving the rendering by a semantic face mask [49, 56]. The methods mostly focused on both modeling and controlling the geometry are typically SDF-based and require learning a separate latent space for expressions [60], oftentimes with the aid of deformations [26, 61]. At the same time, approaches that fit a neural representation to a "talking head"-style video, enable explicit articulation by introducing the parametric model guidance [27, 64, 65].

Inspired by the combination of these ideas, in this research, we introduce a generative model that allows for animation and tracking and preserves high level of detail. At the heart of our approach lies the idea of combining an explicit parametric head model (FLAME [36]) with surface displacements complementing the low geometry detail of the head model. FLAME is an example of a 3D Morphable Model [10, 20] with a fixed set of vertices and fixed topology, constructed as a linear statistical model over the heads with point-to-point correspondence and further controlled by shape and expression latent codes. To obtain the necessary training data, we register a highly subdivided FLAME mesh template with free vertex displacements to all 3D head scans in the NPHM [26] dataset. To facilitate as high level of detail in the displacements as possible, they are fitted in two steps. First, the optimization problem is solved for vector displacements with strong regularization that penalizes very hard for self-intersections of the deformed mesh regions. Afterwards, a separate optimization step refines the displacements only along the normals of the deformed vertices, while keeping the regularization weight low. These displacements are baked into a predefined UV layout. Finally, we train a StyleGAN2 [31] model to generalize over this set of baked 2D displacement maps. This novel architecture allows us to operate at a resolution higher than the conventional FLAME template, enabling the generation of highly detailed and animatable 3D head models.

To validate the efficacy and practical utility of our approach, we evaluate it in several settings. The diversity and fidelity of the generated 3D head meshes is quantitatively and visually compared to other methods w.r.t. the real head scans from the FaceVerse dataset [55], both in UV space and rendered image space. We also explore the applicability of our approach in fitting the latent representation of the generative model to complete or incomplete point cloud data and demonstrate its animation and manipulation capabilities.

Our contributions are as follows:

- We introduce a two-stage registration procedure to craft high-detail displacement maps on top of 3DMMs from 3D scanning data.
- We propose a generative model over displacement maps to not only enhance the low-frequency geometry of FLAME with details but also extend its shape space to all kinds of hairstyle variations.
- We demonstrate the versatility of our method through unconditional sampling, interpolation, semantic hair transfer, and conditioning by depth map or a complete scan.

2. Related Work

Many recent solutions to computer vision problems involving human bodies are built on statistical body models. They are the foundation for building personalized avatars [4, 5, 25, 27, 66], motion tracking [22, 51], scan registration [26], controlling image synthesis [50], and many more. Their line of research divides into two major branches.

Mesh-based Models. Pioneering work in the field [10] proposed 3D morphable models (3DMMs) for identity, expression, and appearance representation of human faces. Their model is built around a 3D template mesh and linear parametric blendshapes derived from PCAs over scan data. With new datasets and registration procedures, their work has been extended from faces to heads [35, 36, 42], hands [46], full-bodies [37], or combinations of these [41, 57]. The template mesh has a fixed topology which provides consistent UV unwrapping and enables fitting to know surface correspondences, e.g. semantic regions and landmarks. Yet, it limits the representative power w.r.t. the overall shape and the level of detail beyond what is provided by the template. Downstream approaches compensate this by optimizing displacements [4, 5, 12, 27, 32, 35, 59] or additional implicit geometry [13, 23] on top of the mesh. Displacements are applied either per-vertex individually [4, 5, 27, 32] or as a displacement map over the whole surface using the consistent UV unwrapping of the template [35, 59]. Our method follows this idea by learning a generative model over displacements maps while exploiting the animation model and surface correspondences of the underlying 3DMM. We demonstrate that learning displacements maps on top of a highly-subdivided template allows to model fine details and, compared to previous works, to introduce significant shape variations.

Implicit Models. The recent success of implicit SDFs [40] and neural radiance fields [38] in 3D modeling has also motivated applying them for statistical body models. Most implicit models learn shape and appearance in a canonical reference space [6, 26, 29, 43, 60] or as displacements on top of an existing model [61]. For better generalization and detail preservation, some approaches use a composition of implicit SDFs to model the canonical space [6, 26]. Articulation and animation is modelled either directly in canonical coordinates [29, 61], through implicit deformation fields [26, 60], explicit joints [6] or blendshape deformations bor-

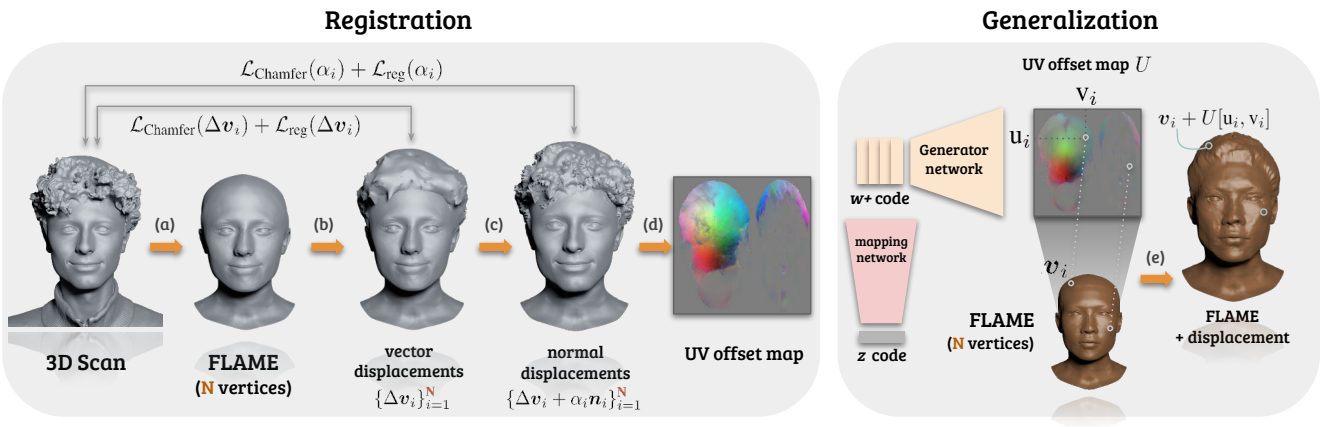

Figure 2. An overview of the method. In the registration stage, we **(a)** fit the FLAME template by the face landmarks to the scan from the NPHM dataset and highly subdivide it, **(b)** optimize for the vertex displacements in $\mathbb{R}^3$ to fit the rough geometry with strong regularizations, **(c)** optimize for the scalar refinements of the displacements along the normal directions, and **(d)** bake the displacements into a UV offset map.To generalize over the UV offset maps, we train a StyleGAN2 [31] model. After training, the offsets can be applied to an arbitrary FLAME template by subdividing it and **(e)** querying the generated UV offset map with the (u, v) locations of the FLAME vertices.

rowed from explicit 3DMMs [56, 64]. While the aforementioned methods rely on multi-view data and aligned 3D scans, a separate line of research demonstrates that statistical shape and appearance priors can also be learned from unstructured image collections [14, 15, 39, 45, 67].

Implicit approaches do not rely on topology and shape templates. This allows them to fit more detail and complex shapes such as hair [26, 60, 61] and even glasses [15, 39]. Yet, it prevents consistent surface correspondences between different samples which need to be explicitly learned [6, 61]. As our approach uses a mesh-based template, it does not suffer from these limitations and has an explicit model for animation. Still, we are able to show that we can provide a comparable level of detail as implicit methods by learning high-resolution displacement maps on top of a highly subdivided template topology.

3. Method

We describe the stages of our method in several parts. Subsec. 3.1 describes the registration procedure for FLAME with surface displacements. Subsec. 3.2 describes learning the generative model. The overview of the approach is presented in Fig. 2.

3.1. Displacements registration procedure

The purpose of this step is to align the FLAME [36] head template with displacements to each scan of the 3D human head dataset. Let us consider a single scanned mesh $\mathcal{P} = (V^{\text{gt}}, \mathcal{F}^{\text{gt}})$ with vertices $V^{\text{gt}} \in \mathrm{R}^{|V^{\text{gt}}|\times 3}$, and faces $\mathcal{F}^{\text{gt}} \in \mathrm{R}^{|\mathcal{F}^{\text{gt}}|\times 3}$.

In order to find the appropriate FLAME parameters for the scan, we follow the rigid alignment optimization procedure outlined in the NPHM work [26]. This procedure requires face landmarks to be known, which can be annotated manually or, as provided with the dataset in our case, calculated via 2D face landmark detectors on the projections of the colored scans and lifted to 3D. This way, we obtain a FLAME template, corresponding to the given scan, and subdivide it via Butterfly algorithm [19]. We will refer to the template after subdivision as to $F = (V, \mathcal{F}, \mathcal{U}_{\mathcal{F}})$, where $V \in \mathrm{R}^{|V|\times 3}$ are the vertices coordinates, $\mathcal{F} \in \mathrm{R}^{|\mathcal{F}|\times 3}$ are the corresponding faces, and $\mathcal{U}_{\mathcal{F}} \in \mathrm{R}^{|\mathcal{F}|\times 3\times 2}$ are the texture coordinates of each vertex in a triangle. Note that using triangle coordinates instead of vertex coordinates is important due to the presence of a seam in the FLAME model, thus making UVs for the seam vertices ambiguous.

As FLAME basis does not represent hair or face details, we define these in a form of vertex displacements and learn them in two stages. During the first stage, we optimize the loss function $\mathcal{L}_{\text{Stage 1}}(D)$ for additive vector displacements $D_{\text{Stage 1}} \in \mathbb{R}^{|V|\times 3}$ of the vertices:

$$\begin{aligned}\mathcal{L}(D, V, \mathcal{F}, V^{\text{gt}} | \boldsymbol{\lambda}) = \lambda^{\text{Chamfer}}\, \mathcal{L}_{\text{Chamfer}}(V + D, V^{\text{gt}}) \\ + \lambda^{\text{edge}}\, \mathcal{L}_{\text{edge}}(V + D, \mathcal{F}) \\ + \lambda^{\text{lapl}}\, \mathcal{L}_{\text{lapl}}(V + D, \mathcal{F})\end{aligned} \tag{1}$$

$$\mathcal{L}_{\text{Stage 1}}(D) = \mathcal{L}(D, V, \mathcal{F}, V^{\text{gt}} \,|\, \boldsymbol{\lambda}_{\text{Stage 1}}) \tag{2}$$

Hyperparameters $\boldsymbol{\lambda}_{\text{Stage 1}} = (\lambda^{\text{Chamfer}}_{\text{Stage 1}}, \lambda^{\text{edge}}_{\text{Stage 1}}, \lambda^{\text{lapl}}_{\text{Stage 1}})$ define the Chamfer matching term weight, the weight of edge length regularization and standard Laplacian regularization. In this stage, the weight of regularizations is high in order to prevent self-intersections that can occur when regressing vector displacements. Also, we only optimize the vector displacements for the hair region.

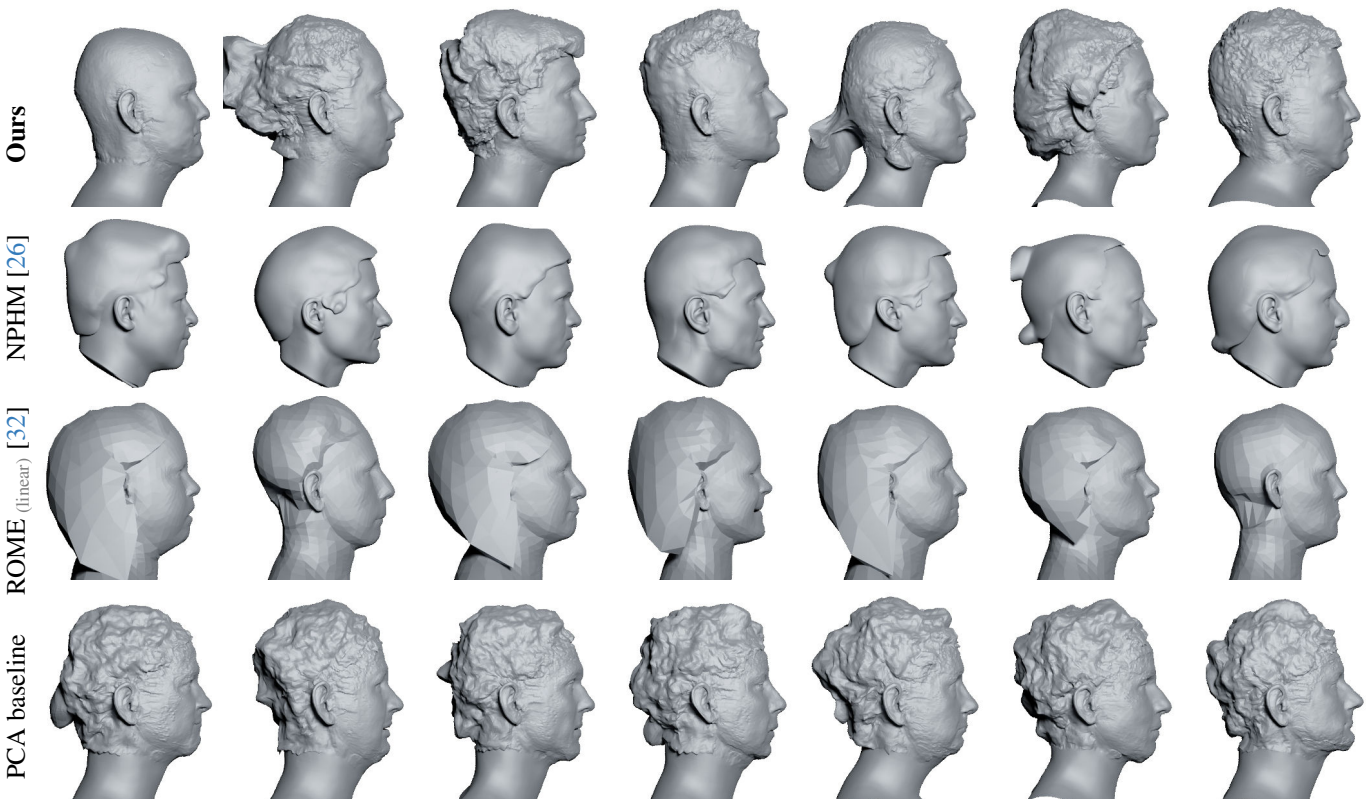

Figure 3. Visual comparison of fidelity and diversity of the meshes generated by various methods. For *Ours*, random FLAMEs are sampled from Gaussian distribution with statistics calculated over the NPHM dataset; same for the *PCA baseline* pre-fitted to our UV registrations. Meshes from *NPHM* are obtained by sampling the latent codes and running marching cubes over the generated SDF representations. We demonstrate higher variability of produced head geometry and better details than the other methods. *Electronic zoom-in recommended.*

In the second stage, we optimize the loss function $\mathcal{L}_{\text{Stage 2}}(\boldsymbol{\alpha})$ for displacements $D_{\text{Stage 2}} \in \mathbb{R}^{|V|\times 3}$ that are only allowed to move over the normals of the previously displaced vertices:

$$D_{\text{Stage 2}} = D_{\text{Stage 1}} + N \odot \boldsymbol{\alpha}, \tag{3}$$

where $N \in \mathbb{R}^{|V|\times 3}$ corresponds to the normals, calculated by numerical difference for vertices deformed after the Stage 1, and $\odot$ defines the element-wise product of rows of N and elements of α (each normal $\boldsymbol{n}_i$ is multiplied by the respective amplitude α_i). $\mathcal{L}_{\text{Stage 2}}(\boldsymbol{\alpha})$ is expressed through the same basic loss expression:

$$\mathcal{L}_{\text{Stage 2}}(\boldsymbol{\alpha}) = \mathcal{L}(D_{\text{Stage 1}} + N \odot \boldsymbol{\alpha}, V, \mathcal{F}, V^{\text{gt}} \,|\, \boldsymbol{\lambda}_{\text{Stage 2}}), \tag{4}$$

while hyperparameters $\boldsymbol{\lambda}_{\text{Stage 2}}$ are selected with relatively lower regularization weights. This allows for fitting high-frequency details while maintaining the same rough shape of the regressed shape. At this stage, we allow both hair and face regions to deform, while subtle parts such as ears and eyeballs are fixed from moving.

Finally, we bake the displacements $D_{\text{Stage 2}}$ into a UV map $U \in \mathbb{R}^{\text{H}\times\text{W}\times 3}$ by rendering it onto the UV space with known texture coordinates $\mathcal{U}_{\mathcal{F}}$ and triangles $\mathcal{F}$.

The registration procedure is repeated for the dataset consisting of multiple 3D scans, resulting in a set of UV displacement maps $(U_1, \ldots, U_S)$.

3.2. Generative model

The described registration procedure allows us to relax the problem of 3D head geometry generation into a problem of generation of 2D UV displacement maps, which allows us to apply a 2D generative model. We have selected StyleGAN2 for that purpose due to its capability of generalizing over relatively small datasets of images [1, 30, 63] while maintaining close-to-SoTA image generation capabilities [31]. The model consists of a mapping network and a generator network, which we will refer to as $f(z)$ together, where $z \in \mathcal{Z} \subset \mathbb{R}^{\text{D}}$ is a latent code sampled from a standard normal distribution during training. The generator produces a UV offset map $U = f(z)$, which we can apply to an arbitrary (anyhow densely subdivided) FLAME template

$F = (V, \mathcal{F}, \mathcal{U}_\mathcal{F})$ by querying the map U with its texture coordinates $\mathcal{U}_\mathcal{F}$ to obtain the respective vertex displacements. We later demonstrate visually that the generated displacements could be applied to an arbitrary FLAME template.

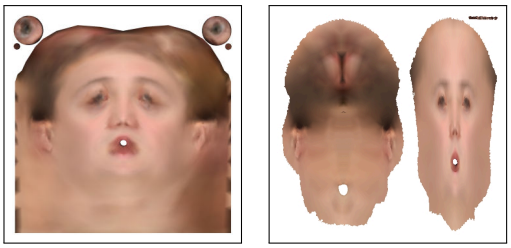

Figure 4. Comparison of the FLAME layouts. The standard, commonly used unwrapping for FLAME *(left)* features a seam corresponding to the vertical line in the back of the head and pays more attention to the facial region than to the scalp. In the hand-crafted custom layout that we employ *(right)*, a different seam around the face border is selected, thus making the regions of face and scalp separated and of similar size, which simplifies modeling complex geometry such as long hair without breaks.

Post-processing. Since the UV map U is generated in the UV layout that contains a seam, we expect StyleGAN to resolve it in general, i.e. produce similar displacements in the face and scalp region near the same seam vertex. Still, there is no dedicated supervision during StyleGAN training that ensures that it always happens and that the border is preserved pixel-perfect. Because of that, before querying the UV map U, we ensure the smoothness of the seam by equalizing the displacements at its borders and blending the displacements in the small vicinity inside and outside of it. The sampling errors near the seam are fixed by filling in all the empty space in the map with the displacements copied from the nearest neighbor pixels in the valid regions.

4. Experiments

4.1. Training procedure

Our method is trained on the private version[2] of the NPHM [26] dataset, consisting of 6975 high-resolution scans of 327 diverse identities captured by two 3D scanners. For the registration procedure, we use Adam optimizer with learning rate of $3 \cdot 10^{-2}$ for the first stage and $3 \cdot 10^{-4}$ for the second stage. The hyperparameters $\boldsymbol{\lambda}_{\text{Stage 1}} = (\lambda^{\text{Chamfer}}_{\text{Stage 1}}, \lambda^{\text{edge}}_{\text{Stage 1}}, \lambda^{\text{lapl}}_{\text{Stage 1}})$ equal to $(2 \cdot 10^3, 2 \cdot 10^5, 10^4)$. For the second stage, $\boldsymbol{\lambda}_{\text{Stage 2}} = (2 \cdot 10^4, 2 \cdot 10^4, 10^4)$. In the Chamfer loss, we additionally apply correspondences pruning by distance of 1.0, which defines that all the correspondences between source and target with the distance more than 1.0 in the NPHM coordinate system are automatically discarded. This has been introduced for more consistent gradual learning of displacements, such that at each optimization step, only the nearest points affect the deformation learning.

For the generative model training, we use StyleGAN2 implementation with all augmentations turned off (since they wouldn't yield valid UV maps in our case), 8 mapping network layers and a high gradient penalty of 4.0. The learning rates are $2 \cdot 10^{-3}$ for the generator and $1 \cdot 10^{-3}$ for the discriminator. We train it for 72K steps with the batch size of 8 and the resolution of the UV maps of 256×256.

	FID ↓	KID ↓	IS ↑	Rel. IS ↑	MMD ↓	JSD ↓	COV ↑
Ours	**72.37**	**0.071**	**1.67**	**88.25%**	**6.45**	21.41	**47.12%**
PCA	102.96	0.125	1.46	77.13%	9.96	20.68	22.12%
NPHM [26]	139.82	0.170	1.57	82.95%	7.80	**19.06**	46.15%
ROME [32]	169.65	0.204	1.63	86.81%	10.02	23.19	32.69%
FLAME [36]	198.85	0.262	1.13	59.86%	12.95	23.89	5.77%

Table 1. The comparison of quality and diversity of random samples generated by each of the methods. FID and KID measure the similarity of the generated mesh renderings vs. the renderings of the ground truth meshes in FaceVerse dataset, while IS measures the realism of the generated renderings. 3D metrics MMD, JSD, COV assess the similarity of point clouds sampled from generated and ground truth meshes. For this and all other tables, MMD has been multiplied by 10^3 and JSD is multiplied by 10^2.

4.2. Results

Unconditional sampling. In Fig. 3, we compare the difference in details and diversity of the unconditional samples produced by our method to the ones produced by NPHM [26] and ROME [32] methods. Additionally, we compare it to the PCA baseline, whereas PCA linear basis is fitted to our UV displacement maps, and provide the numbers for random FLAME samples without added displacements as a reference. For Ours, PCA baseline and ROME, a FLAME with random shape, expression and jaw parameters are sampled from a Gaussian for every head mesh, in accordance with the statistics precalculated over the NeRSemble dataset [34]. While NPHM, PCA baseline, and Ours have been fitted to exactly the same training dataset, for ROME, the authors' provided checkpoint has been used. For ROME, we sample the FLAME displacements from a linear model provided by the authors of ROME as the sampling strategy proposed by the ROME authors. Visually, we observe both higher diversity and better representation of hair details than for all baselines. The details of the facial region are generally the sharpest for ROME, PCA baseline, and Ours, due to the use of the FLAME template.

In Table 1, we also quantify the level of detail and variety of the generated meshes w.r.t. the full head scans from the FaceVerse dataset [55] that has not been used for training. The comparison is performed in two ways. Firstly,

[2]The private version was provided to us by the authors of NPHM [26] and is going to be released to the public. The public version, available at the time of the publication, is roughly a 30% subset of the private version.

to evaluate the visual plausibility of the generated geometry, we render 2195 ground truth meshes from FaceVerse and the same number of meshes generated by each method with highly metallic material from eight distinct viewpoints, uniformly sampled along the circular trajectory in the horizontal plane. The **FID** [28] and **KID** [9] perceptual metrics are calculated for all generated and ground truth renderings from a given viewpoint and then macro-averaged over eight viewpoints. Inception Score (**IS**) [47] evaluated the realism of the generated renderings. Because of that, we also report the relative IS as a percentage of the reference IS calculated for the ground truth renderings.

Secondly, we compare the distributions of point clouds sampled from the generated and ground truth meshes. To do that, we sample 10K points from each of the 2195 generated and the same number of ground truth meshes and calculate several 3D similarity metrics. Jensen-Shannon Divergence (**JSD**) is evaluated by comparing the distributions of generated and ground truth points, splat into a voxel grid (in our case, of 512^3 voxels). Minimum Matching Distance (**MMD**) is a measure of 3D object realism that, for each ground truth sample, involves evaluating the distance to the most similar sample in the generated set. Similarly, Coverage (**COV**) indicates the percentage of the ground truth samples, for which the nearest neighbor among all ground truth and generated samples falls into the generated set. The detailed mathematical description of each of the 3D metrics we report can be found in [21, 58]. In addition, we demonstrate how much the generated samples deviate from the NPHM training set in Fig. 5. The nearest neighbor scan is found by comparing the generated UV map to the UV maps comprised of registered ground truth displacements for all training scans by L2 distance over the scalp region.

The results in Table 1 indicate that the renderings from our method appear more realistic than of the other methods, with either PCA baseline or NPHM performing similar according to different subsets of the metrics. Close promixity to NPHM by MMD, COV, JSD could be explained by training on exactly the same dataset.

Ablating over the choice of the UV layout. We assess the effect of a manually hand-crafted UV space for FLAME on the quality of generations in Fig. 6. As observed, moving the seam from the vertical middle line, as in the standard UV layout for FLAME, to the face border, allows us to model more consistent and complex geometry without large distinction between a left and a right part.

Ablating over the choice of the generative model architecture. We compare StyleGAN to other state-of-the-art generative model architectures, namely of VAE [33] and VQ-VAE [53] family, with ResNet-18 encoder and decoder. For VQ-VAE, the sampling from the latent space is implemented via training PixelCNN autoregressive model [52]. The results are presented in Table 2 and in Fig. 7.

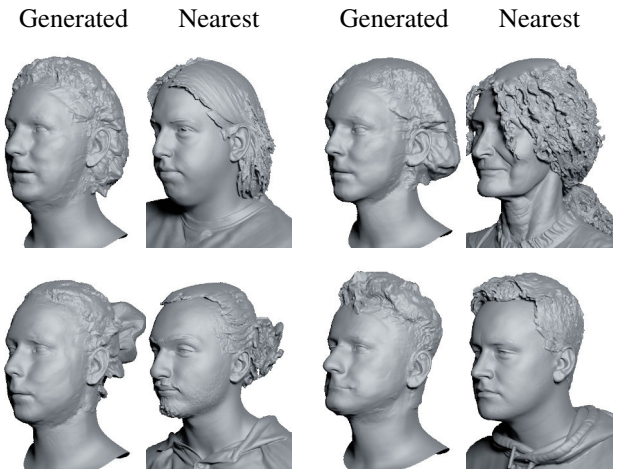

Figure 5. Randomly generated samples from HeadCraft and the corresponding nearest neighbors in the NPHM dataset among the scans used for training.

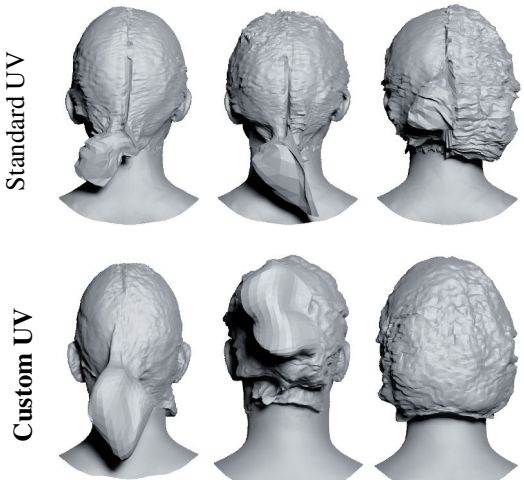

Figure 6. Ablation over the choice of the UV layout. Our method utilizes custom UV layout that allows us to model more consistent geometry by mitigating seam artifacts, as seen from the back view here. Layouts are shown in Fig. 4.

	FID ↓	KID ↓	IS ↑	Rel. IS ↑	MMD ↓	JSD ↓	COV ↑
Ours	**72.37**	**0.07**	**1.67**	**88.25%**	**6.45**	21.41	**47.12%**
VAE [33]	108.91	0.13	1.66	87.79%	6.79	21.65	42.31%
VQ-VAE [53]	125.65	0.15	1.62	85.94%	7.96	21.88	38.46%
PCA	102.96	0.13	1.46	77.13%	9.96	**20.68**	22.12%

Table 2. Ablation over the generative model design. VAE and VQ-VAE follow the ResNet-18 encoder and decoder architecture, while *Ours* is based on StyleGAN2. We also include PCA baseline scores here as a reference.

Behavior of the registration procedure. In Fig. 10, we demonstrate the advantage of the two-stage registration procedure, described in Subsec. 3.1, over omitting one of the stages. As can be seen, keeping only the vector displace-

ment optimization results in too rough shape, and relaxing the regularization constraints yields significant artifacts such as self-intersections and spikes. Running the normal displacement stage without any preliminary vector displacement stage performs similarly to our two-stage procedure but produces artifacts for long hair that does not trivially project onto the surface. In turn, it can produce the mappings between template vertices and scan vertices, inconsistent across various samples for the long hair parts.

4.3. Applications

Fitting the latent code to a depth map. Our model can act as a prior for completing the partial observations, e.g. when they come from a depth sensor. To evaluate the performance of the model in that scenario, we demonstrate the completion capabilities of the model over a number of scans from NPHM corresponding to the subjects unseen during training. For each of these scans, we project their depth onto random viewpoints in the frontal hemisphere and project it back to 3D to construct partial point clouds. To obtain a partial UV map to be completed, we run our registration procedure with a few modifications to fit a part of the scan. Namely, we only fit the points within the convex hull of the partial point cloud, apply stronger edge length regularization weight, and constrain the points at the border of the allowed region from moving. The final mask of observed UV texels is refined by only selecting those points that turn out to be close to the partial point cloud. Finally, a latent code of HeadCraft explaining the partial UV map is found via StyleGAN inversion techniques. More technical details of the partial registration and inversion are provided in the Supplementary [3]. The fitting quality can be evaluated by the visual comparison in Fig. 8.

Figure 7. Ablation over the generative model design. VAE and VQ-VAE both follow the ResNet-18 encoder and decoder architecture, while our method is based on StyleGAN2. The results from VAE and VQ-VAE match the diversity of the training data but not the level of detail and handle the UV seam worse.

The capabilities of fitting the model to the full scan, e.g. created from Structure-from-Motion (SfM), are demonstrated as a part of the semantic editing experiments in Fig. 11 (top rows, the result of the latent fitting to each of the scans, $\lambda = \{0, 1\}$).

Animation. The decomposition of the parametric model and the displacements allows us to animate the complete head model. In our experiments, we take real multi-view video sequences with talking people from the NeRSemble dataset [34] and obtain shape, expression, jaw, and head pose parameters for each time frame of the speech by running a FLAME tracker for each sequence. For each of the sample shapes, estimated from the sequences, we reenact the corresponding FLAME with estimated expression parameters, subdivide the template and query a randomly presampled UV displacement map from HeadCraft. Since the template is also deforming over time, we rotate the displacements according to the changing surface normals of the template. More results are demonstrated in Fig. 9 and in the Supplementary Video.

Figure 9. Demonstration of the animation capabilities of the model. Each of the sequences is created by taking FLAME shape parameters, expression, jaw, and head pose parameters from a sequence from NeRSemble dataset [34], subdividing the template, and applying randomly generated displacements from HeadCraft.

4.4. Analysis

Interpolation between the displacements. In Fig. 11, we show how interpolating the latent code of our generative model influences the change of the geometry. Further interpolations are presented in the Supplementary Video.

Hair transfer from one scan to another. Access to the shared UV space allows us to modify the geometry semantically. In Fig. 11, the transfer of the scalp region from one ground truth NPHM scan, unseen during training, to the other is shown. The transfer is performed via fitting the latent representation of HeadCraft to the driver scan (the source of displacements) and feeding it to the model. The extracted displacements are later applied to the source scan.

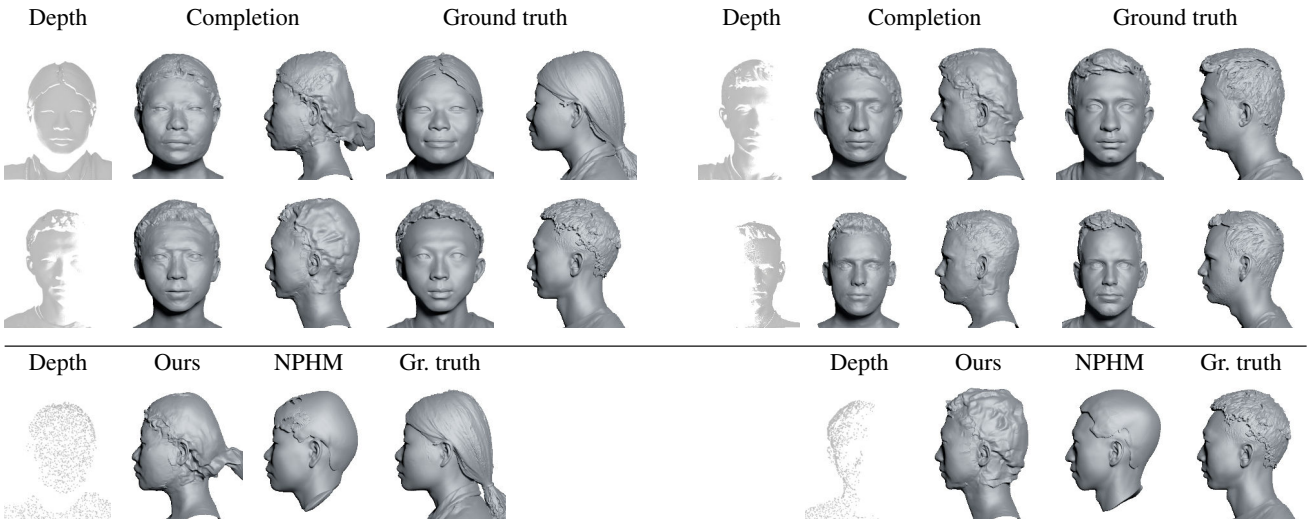

Figure 8. Demonstration of geometry completion aided by the HeadCraft model. Here, we extract depth maps from scans from the NPHM dataset, unseen during training, and try to complete them by finding the appropriate latent representation of StyleGAN. As a necessary intermediate step, we first apply our registration procedure to the partial point cloud to locate the points in the UV space of the template. The optimal latent is found by minimizing the discrepancy of the complete UV map and registered partial UV map in the observed regions. HeadCraft is also capable of estimating plausible details for a very sparse point cloud (1% of # points) – see the last row.

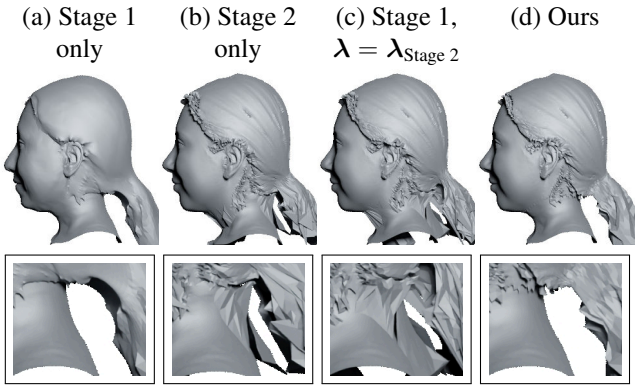

Figure 10. Ablation over the one-stage vs. two-stage registration. Leaving only the vector displacements stage (a) yields too smooth geometry, and learning them only along the normals (b) introduces unnecessary spikes – just like running the first stage with weaker regularization (c).

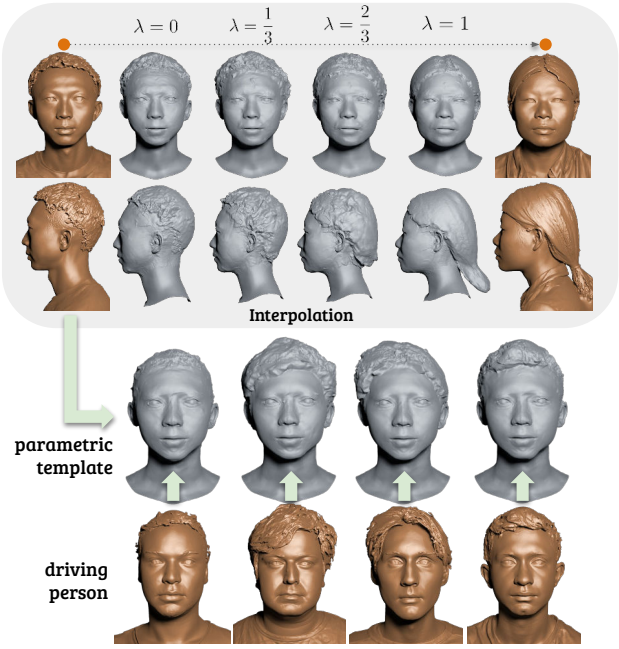

Figure 11. Semantic editing. Interpolating the latent representations of HeadCraft allows us to smoothly change the person's appearance from one to another. To do that, we fit the latent codes for two real scans in the NPHM dataset (brown, top rows), unseen during training, and blend them together with a λ weight. Likewise, we can transfer hair geometry from one person to another.

5. Discussion

In this work, a generative model for 3D human heads is presented. We demonstrate the efficacy of the hybrid approach involving an underlying animatable parametric model and a neural vertex displacement modeller. Most importantly, our method allows to model high-quality shape variations while maintaining the realistic animation capability, and the inversion framework allows us to find a suitable latent representation to either represent a full head scan or a part of it that could come from e.g. the depth sensor. A direction of the possible future work could be focused on incorporating an appearance model for color and material-based relighting and a physical model of hair movement, based on, for instance, hair strands, to support more realistic animation.

Acknowledgments. We gratefully acknowledge the support of this research by a TUM-IAS Hans Fischer Senior Fellowship, the ERC Starting Grant Scan2CAD (804724) and the Horizon Europe vera.ai project (101070093). We also thank Yawar Siddiqui and Alexey Artemov for helpful advice, Tobias Kirschstein for his assistance with the NeRSembles dataset and visualization, Taras Khakhulin for his help with the ROME baseline, Peter Kocsis, Antonio Allegro, Jiapeng Tang for early peer review, Silvia Sellán for the Blender visualization course materials that we used, and Angela Dai for the video voiceover.

References

[1] GitHub: Awesome Pretrained Stylegan by justinpinkney. A collection of pre-trained StyleGAN models trained on different datasets at different resolution. https://github.com/justinpinkney/awesome-pretrained-stylegan. 4

[2] Learn OpenGL – Normal Mapping. https://learnopengl.com/Advanced-Lighting/Normal-Mapping. 16

[3] HeadCraft: Modeling High-Detail Shape Variations for Animated 3DMMs. Supplementary Material. 7

[4] Thiemo Alldieck, Marcus Magnor, Weipeng Xu, Christian Theobalt, and Gerard Pons-Moll. Detailed human avatars from monocular video. In *2018 International Conference on 3D Vision (3DV)*, pages 98–109. IEEE, 2018. 2

[5] 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 *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1175–1186, 2019. 2

[6] Thiemo Alldieck, Hongyi Xu, and Cristian Sminchisescu. imGHUM: Implicit generative models of 3D human shape and articulated pose. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 5461–5470, 2021. 2, 3

[7] Sizhe An, Hongyi Xu, Yichun Shi, Guoxian Song, Umit Ogras, and Linjie Luo. Panohead: Geometry-aware 3d full-head synthesis in 360. *arXiv preprint arXiv:2303.13071*, 2023. 2

[8] Alexander Bergman, Petr Kellnhofer, Wang Yifan, Eric Chan, David Lindell, and Gordon Wetzstein. Generative neural articulated radiance fields. *Advances in Neural Information Processing Systems*, 35:19900–19916, 2022. 2

[9] Mikołaj Bińkowski, Danica J Sutherland, Michael Arbel, and Arthur Gretton. Demystifying mmd gans. *arXiv preprint arXiv:1801.01401*, 2018. 6

[10] Volker Blanz and Thomas Vetter. A Morphable Model for the Synthesis of 3D faces. In *Proceedings of the 26th Annual Conference on Computer Graphics and Interactive Techniques*, page 187–194. ACM Press/Addison-Wesley Publishing Co., 1999. 2

[11] Andrew Brock, Jeff Donahue, and Karen Simonyan. Large scale gan training for high fidelity natural image synthesis. *arXiv preprint arXiv:1809.11096*, 2018. 13

[12] Andrei Burov, Matthias Nießner, and Justus Thies. Dynamic surface function networks for clothed human bodies. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 10754–10764, 2021. 2

[13] Chen Cao, Tomas Simon, Jin Kyu Kim, Gabe Schwartz, Michael Zollhoefer, Shun-Suke Saito, Stephen Lombardi, Shih-En Wei, Danielle Belko, Shoou-I Yu, et al. Authentic volumetric avatars from a phone scan. *ACM Transactions on Graphics (TOG)*, 41(4):1–19, 2022. 2

[14] Eric R Chan, Marco Monteiro, Petr Kellnhofer, Jiajun Wu, and Gordon Wetzstein. pi-GAN: Periodic implicit generative adversarial networks for 3d-aware image synthesis. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 5799–5809, 2021. 1, 3

[15] Eric R Chan, Connor Z Lin, Matthew A Chan, Koki Nagano, Boxiao Pan, Shalini De Mello, Orazio Gallo, Leonidas J Guibas, Jonathan Tremblay, Sameh Khamis, et al. Efficient geometry-aware 3D generative adversarial networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 16123–16133, 2022. 1, 3

[16] Yuhao Cheng, Yichao Yan, Wenhan Zhu, Ye Pan, Bowen Pan, and Xiaokang Yang. Head3d: Complete 3d head generation via tri-plane feature distillation. *arXiv preprint arXiv:2303.15892*, 2023. 2

[17] Paolo Cignoni, Marco Callieri, Massimiliano Corsini, Matteo Dellepiane, Fabio Ganovelli, and Guido Ranzuglia. MeshLab: an Open-Source Mesh Processing Tool. In *Eurographics Italian Chapter Conference*. The Eurographics Association, 2008. 12

[18] Blender Online Community. *Blender - a 3D modelling and rendering package*. Blender Foundation, Stichting Blender Foundation, Amsterdam, 2018. 12

[19] Nira Dyn, David Levine, and John A Gregory. A butterfly subdivision scheme for surface interpolation with tension control. *ACM transactions on Graphics (TOG)*, 9(2):160–169, 1990. 3

[20] Bernhard Egger, William AP Smith, Ayush Tewari, Stefanie Wuhrer, Michael Zollhoefer, Thabo Beeler, Florian Bernard, Timo Bolkart, Adam Kortylewski, Sami Romdhani, et al. 3d morphable face models—past, present, and future. *ACM Transactions on Graphics (ToG)*, 39(5):1–38, 2020. 2

[21] Ziya Erkoç, Fangchang Ma, Qi Shan, Matthias Nießner, and Angela Dai. HyperDiffusion: Generating implicit neural fields with weight-space diffusion. *arXiv preprint arXiv:2303.17015*, 2023. 6

[22] 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(4): 1–13, 2021. 2

[23] Yao Feng, Weiyang Liu, Timo Bolkart, Jinlong Yang, Marc Pollefeys, and Michael J. Black. Learning disentangled avatars with hybrid 3d representations. *arXiv*, 2023. 2

[24] Steven Fortune. Voronoi diagrams and delaunay triangulations. In *Handbook of discrete and computational geometry*, pages 705–721. Chapman and Hall/CRC, 2017. 13

[25] 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)*, pages 8649–8658, 2021. 2

[26] Simon Giebenhain, Tobias Kirschstein, Markos Georgopoulos, Martin Rünz, Lourdes Agapito, and Matthias Nießner. Learning neural parametric head models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 21003–21012, 2023. 2, 3, 4, 5

[27] 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*, pages 18653–18664, 2022. 2

[28] 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. *Advances in neural information processing systems*, 30, 2017. 6

[29] Yang Hong, Bo Peng, Haiyao Xiao, Ligang Liu, and Juyong Zhang. Headnerf: A real-time nerf-based parametric head model. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 20374–20384, 2022. 2

[30] Tero Karras, Miika Aittala, Janne Hellsten, Samuli Laine, Jaakko Lehtinen, and Timo Aila. Training generative adversarial networks with limited data. *Advances in neural information processing systems*, 33:12104–12114, 2020. 4

[31] Tero Karras, Samuli Laine, Miika Aittala, Janne Hellsten, Jaakko Lehtinen, and Timo Aila. Analyzing and improving the image quality of stylegan. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 8110–8119, 2020. 2, 3, 4

[32] Taras Khakhulin, Vanessa Sklyarova, Victor Lempitsky, and Egor Zakharov. Realistic one-shot mesh-based head avatars. In *European Conference on Computer Vision*, pages 345–362. Springer, 2022. 2, 4, 5

[33] Diederik P Kingma, Max Welling, et al. An introduction to variational autoencoders. *Foundations and Trends® in Machine Learning*, 12(4):307–392, 2019. 6

[34] Tobias Kirschstein, Shenhan Qian, Simon Giebenhain, Tim Walter, and Matthias Nießner. NeRSemble: Multi-view Radiance Field Reconstruction of Human Heads. *arXiv preprint arXiv:2305.03027*, 2023. 5, 7

[35] Ruilong Li, Karl Bladin, Yajie Zhao, Chinmay Chinara, Owen Ingraham, Pengda Xiang, Xinglei Ren, Pratusha Prasad, Bipin Kishore, Jun Xing, et al. Learning formation of physically-based face attributes. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 3410–3419, 2020. 2

[36] Tianye Li, Timo Bolkart, Michael J Black, Hao Li, and Javier Romero. Learning a model of facial shape and expression from 4D scans. *ACM Trans. Graph.*, 36(6):194–1, 2017. 2, 3, 5, 12

[37] Matthew Loper, Naureen Mahmood, Javier Romero, Gerard Pons-Moll, and Michael J. Black. SMPL: A skinned multi-person linear model. *ACM Trans. Graphics (Proc. SIGGRAPH Asia)*, 34(6):248:1–248:16, 2015. 2

[38] 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. *CoRR*, abs/2003.08934, 2020. 1, 2

[39] Roy Or-El, Xuan Luo, Mengyi Shan, Eli Shechtman, Jeong Joon Park, and Ira Kemelmacher-Shlizerman. Stylesdf: High-resolution 3d-consistent image and geometry generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13503–13513, 2022. 2, 3

[40] Jeong Joon Park, Peter Florence, Julian Straub, Richard Newcombe, and Steven Lovegrove. DeepSDF: Learning continuous signed distance functions for shape representation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 165–174, 2019. 1, 2

[41] Georgios Pavlakos, Vasileios Choutas, Nima Ghorbani, Timo Bolkart, Ahmed A. A. Osman, Dimitrios Tzionas, and Michael J. Black. Expressive body capture: 3D hands, face, and body from a single image. In *Proceedings IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 10975–10985, 2019. 2

[42] Stylianos Ploumpis, Evangelos Ververas, Eimear O'Sullivan, Stylianos Moschoglou, Haoyang Wang, Nick Pears, William AP Smith, Baris Gecer, and Stefanos Zafeiriou. Towards a complete 3d morphable model of the human head. *IEEE transactions on pattern analysis and machine intelligence*, 43(11):4142–4160, 2020. 2

[43] Eduard Ramon, Gil Triginer, Janna Escur, Albert Pumarola, Jaime Garcia, Xavier Giro-i Nieto, and Francesc Moreno-Noguer. H3d-net: Few-shot high-fidelity 3d head reconstruction. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 5620–5629, 2021. 2

[44] 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. 12

[45] Daniel Rebain, Mark Matthews, Kwang Moo Yi, Dmitry Lagun, and Andrea Tagliasacchi. Lolnerf: Learn from one look. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1558–1567, 2022. 3

[46] Javier Romero, Dimitrios Tzionas, and Michael J. Black. Embodied hands: Modeling and capturing hands and bodies together. *ACM Transactions on Graphics, (Proc. SIGGRAPH Asia)*, 36(6), 2017. 2

[47] Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. Improved techniques for training gans. *Advances in neural information processing systems*, 29, 2016. 6

[48] Tilo Strutz. The distance transform and its computation. *arXiv preprint arXiv:2106.03503*, 2021. 13

[49] Junshu Tang, Bo Zhang, Binxin Yang, Ting Zhang, Dong Chen, Lizhuang Ma, and Fang Wen. Explicitly controllable 3D-aware portrait generation. *arXiv preprint arXiv:2209.05434*, 2022. 2

[50] Ayush Tewari, Mohamed Elgharib, Gaurav Bharaj, Florian Bernard, Hans-Peter Seidel, Patrick Pérez, Michael Zoll-

hofer, and Christian Theobalt. Stylerig: Rigging stylegan for 3d control over portrait images. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6142–6151, 2020. 2

[51] Justus Thies, Michael Zollhofer, Marc Stamminger, Christian Theobalt, and Matthias Nießner. Face2face: Real-time face capture and reenactment of rgb videos. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2387–2395, 2016. 2

[52] Aäron Van Den Oord, Nal Kalchbrenner, and Koray Kavukcuoglu. Pixel recurrent neural networks. In *International conference on machine learning*, pages 1747–1756. PMLR, 2016. 6

[53] Aaron Van Den Oord, Oriol Vinyals, et al. Neural discrete representation learning. *Advances in neural information processing systems*, 30, 2017. 6

[54] Jörg Vollmer, Robert Mencl, and Heinrich Mueller. Improved laplacian smoothing of noisy surface meshes. In *Computer graphics forum*, pages 131–138. Wiley Online Library, 1999. 16

[55] Lizhen Wang, Zhiyuan Chen, Tao Yu, Chenguang Ma, Liang Li, and Yebin Liu. Faceverse: a fine-grained and detail-controllable 3D face morphable model from a hybrid dataset. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 20333–20342, 2022. 2, 5

[56] Yue Wu, Yu Deng, Jiaolong Yang, Fangyun Wei, Qifeng Chen, and Xin Tong. Anifacegan: Animatable 3d-aware face image generation for video avatars. *Advances in Neural Information Processing Systems*, 35:36188–36201, 2022. 2, 3

[57] Hongyi Xu, Eduard Gabriel Bazavan, Andrei Zanfir, William T. Freeman, Rahul Sukthankar, and Cristian Sminchisescu. Ghum & ghuml: Generative 3d human shape and articulated pose models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020. 2

[58] Guandao Yang, Xun Huang, Zekun Hao, Ming-Yu Liu, Serge Belongie, and Bharath Hariharan. Pointflow: 3D point cloud generation with continuous normalizing flows. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 4541–4550, 2019. 6, 13

[59] Haotian Yang, Hao Zhu, Yanru Wang, Mingkai Huang, Qiu Shen, Ruigang Yang, and Xun Cao. Facescape: a large-scale high quality 3d face dataset and detailed riggable 3d face prediction. In *Proceedings of the ieee/cvf conference on computer vision and pattern recognition*, pages 601–610, 2020. 2

[60] Tarun Yenamandra, Ayush Tewari, Florian Bernard, Hans-Peter Seidel, Mohamed Elgharib, Daniel Cremers, and Christian Theobalt. i3DMM: Deep implicit 3D morphable model of human heads. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12803–12813, 2021. 2, 3

[61] Mihai Zanfir, Thiemo Alldieck, and Cristian Sminchisescu. PhoMoH: Implicit Photorealistic 3D Models of Human Heads. *arXiv preprint arXiv:2212.07275*, 2022. 2, 3

[62] Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 586–595, 2018. 14

[63] Shengyu Zhao, Zhijian Liu, Ji Lin, Jun-Yan Zhu, and Song Han. Differentiable augmentation for data-efficient gan training. *Advances in neural information processing systems*, 33:7559–7570, 2020. 4

[64] 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 *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13545–13555, 2022. 2, 3

[65] Yufeng Zheng, Wang Yifan, Gordon Wetzstein, Michael J Black, and Otmar Hilliges. PointAvatar: Deformable point-based head avatars from videos. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 21057–21067, 2023. 2

[66] Wojciech Zielonka, Timo Bolkart, and Justus Thies. Instant volumetric head avatars. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4574–4584, 2023. 2

[67] Jinlong Yang Michael J.Black Otmar Hilliges Andreas Geiger Zijian Dong, Xu Chen. AG3D: Learning to generate 3D avatars from 2D image collections. In *International Conference on Computer Vision (ICCV)*, 2023. 3

A. Method: Technical Details

A.1. Displacement registration procedure

Here we explain the procedure in more detail. The vertices and displacements are modeled in the NPHM coordinate system, aligned with the scans, and the scaling of $30\times$ applied. The implementation of Butterfly subdivision of the FLAME template from MeshLab [17] was used. The parameters of the subdivision are constant across the scans and equal to 3 subdivision iterations with a threshold of 42.5. The subdivision produces around 100K vertices and 200K triangles for the original template consisting of 5023 vertices and 9976 triangles and smooths the surface. The description of the optimization problem features individual loss terms. The expanded expression for the terms are as follows. The Chamfer term $\mathcal{L}_{\text{Chamfer}}(P_1, P_2)$ quantifies the distance between the point clouds $P_1 \in \mathbb{R}^{|P_1|\times 3}$ and $P_2 \in \mathbb{R}^{|P_2|\times 3}$ is supposed to be differentiable w.r.t. the points of P_1. In our work, we apply the version pruned by the distance of the correspondences, i.e. when the Euclidean distance between point and its matched version exceeds the predefined threshold d, this correspondence is not accounted in the loss term.

$$
\begin{aligned}
&\mathcal{L}_{\text{Chamfer}}(P_1, P_2) = \\
&\quad \frac{1}{\sum_{p\in P_1}[d(p, nn(p, P_2)) \leq d]} \cdot \\
&\quad \cdot \sum_{p\in P_1} \left(d(p, nn(p, P_2)) \cdot [d(p, nn(p, P_2)) \leq d]\right) \\
&\quad + \frac{1}{\sum_{p\in P_2}[d(p, nn(p, P_1)) \leq d]} \cdot \\
&\quad \cdot \sum_{p\in P_2} \left(d(p, nn(p, P_1)) \cdot [d(p, nn(p, P_1)) \leq d]\right),
\end{aligned}
$$

where $d(\cdot, \cdot)$ stands for the Euclidean distance between two points in space and $nn(p, P) = \arg\min_{p'\in P} d(p, p')$ is the nearest neighbor of p in a point cloud P.

Edge length regularization is defined as follows.

$$
\mathcal{L}_{\text{edge}}(V, \mathcal{F}) = \frac{1}{|E|} \sum_{(e_a, e_b)\in E} d^2(V_{e_a}, V_{e_b}), \tag{5}
$$

where $E = E(\mathcal{F})$ is a set of graph edges derived from the faces $\mathcal{F}$. To construct it, we consider each face bringing three new edges and later leave only the unique edges in E.

Laplacian term is defined as the Euclidean distance between the vertex and its neighbors, which can be efficiently calculated via computing sparse Laplacian $L = L(V, \mathcal{F})$ of the graph:

$$
\mathcal{L}_{\text{lapl}}(V, \mathcal{F}) = \sqrt{\sum_{v\in V} \|Lv\|_2^2} \tag{6}
$$

The outer norm is used instead of e.g. L1 averaging to enforce the uniform smoothness of the mesh and avoid spikes that tend to appear otherwise (see, e.g., the documented example in PyTorch3D [44] repository).

During the vector displacement stage, only the scalp region (defined by the semantic mask shipped with FLAME [36]) is optimized. During the normal displacement stage, we also unfreeze the facial region but keep the neck, eyeballs, ears and inner mouth region frozen (the latter is annotated manually in Blender [18] package and is frozen because of its absence in the ground truth scans, as it is placed fully interior). Each stage takes around 3 min for 1K steps on a single NVIDIA RTX 2080 Ti GPU. We used the PyTorch3D [44] functions for implementation of all the loss terms.

In Fig. 12, we show how the seam was annotated for the custom UV layout. The Blender [18] 4.0 package was used for annotation.

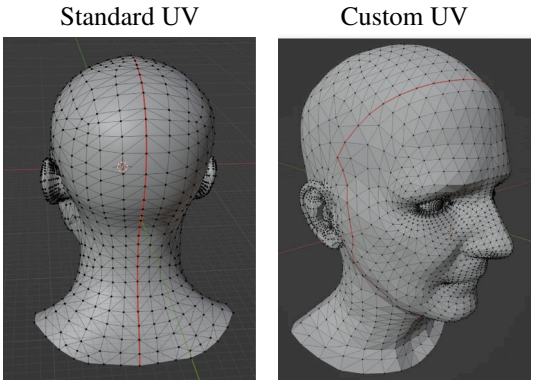

Figure 12. Demonstration of the custom UV seam annotation. *Standard UV* corresponds to the UV space shipped with the FLAME model. *Custom UV* stands for the handcrafted layout employed in our pipeline. This layout simplifies learning sophisticated shape variations such as long hair in a consistent way.

The displacement vectors entries typically belong to the [-2, +2] range, while some large shape variations (e.g. a pony tail) can introduce the offsets into a large range up to [-20, +20]. We clip any displacements, obtained after full registration, to the [-20, +20] range. As the last step, the displacements are rendered in UV space, and each UV map is saved as uint16 image files linearly renormalized from [-20, +20] to $[0, 2^{16} - 1]$. Saving in uint16 (double-byte intensity value) instead of the widely used uint8 (single-byte intensity value) is important, since most of the displacements vector entries are concentrated around the small neighborhood of zero and the precision can be lost when renormal-

izing from [-20, +20] to $[0, 2^8 - 1]$ instead and discretizing. Saving UV maps as raw files would otherwise facilitate much slower training of the generative model due to the time-consuming loading and memory usage overhead.

A.2. Generative model

For training StyleGAN, we used the stylegan2-ada-lightning implementation with ADA and augmentations turned off and the following hyperparameters:

latent dim	# layers (z $\rightarrow$ w)	G lr	D lr
512	8	0.002	0.001

λ_{gp}	λ_{plp}	img size	batch size
4.0	2.0	256×256	8

The model was trained on four NVIDIA RTX 2080 Ti GPUs. For training of the StyleGAN, we replaced the facial part of the registered UV maps of subjects in the NPHM dataset with the corresponding facial part from the neutral expression scan of the same person. This has been done to better support various expressions We also found it beneficial to disable the StyleGAN noise, typically injected into the generator, for the face part of the UV map, to smooth out the generated facial displacements relative to the scalp displacements that normally require higher level of detail. For the scalp region, the StyleGAN noise is constant, initialized separately for each generator layer before training.

Post-processing. The non-smoothness of the seam is mitigated in two ways, which are described in more detail here. The first step is blending the displacements. To do that, we analyze the UV coordinates of the triangles in the custom layout $\mathcal{U}_{\mathcal{F}}$ and find the triangles, for which at least one of its vertices lies on the seam. These are the triangles, for which some of the UV coordinates belong to the left part of the UV map (scalp) and some belong to the right part (face). This also provides us with the texel (i.e. UV map pixel) correspondence between the seam vertices. To softly blend the vertices inside the scalp region (left part of the UV map) near the seam, we first copy the displacements from the seam texels on the right to the respective texels on the left. Then, we compute the coordinates of the respective nearest seam texel from all texels inside the scalp region by applying distance transform algorithm [48]. In the vicinity of 10 texels from the seam (in any direction inside the region), we softly blend the displacements between the original and the seam ones:

$$U[i,j] := \frac{10 - \text{dist\_to\_seam}[i,j]}{10} U[\text{nn}_i[i,j], \text{nn}_j[i,j]] + \frac{\text{dist\_to\_seam}[i,j]}{10} U[i,j], \quad \text{for all } (i,j) \text{ in the 10-texel vicinity of the seam,}$$

where $(\text{nn}_i[i,j], \text{nn}_j[i,j])$ stands for the closest found seam texel to (i,j) and $\text{dist\_to\_seam}[i,j]$ stands for the distance to it. The same procedure to applied vice versa: the seam displacements from the left are copied to the face region border and blended softly inside the face region.

The second step is aimed to fix the sampling errors that can occur when sampling a generated UV map. To do that, we again calculate the distance transform, this time for the empty space in the UV map (regions corresponding to no useful information). This way, the nearest to the seam texel is found for each texel of the empty space, and we copy the displacement from the nearest seam texel into this texel. Effectively, this replaces the empty space with the Voronoi diagram [24] of proximity to the seam, with displacements from the seam occupying each region of the diagram. As a result, when the displacements are being sampled from the UV map with the empty space filling procedure applied in advance, no UV coordinates fall into the space where no displacements are set. In case of hitting the space that was previously empty, the displacement attains a value close to the nearest displacement at the seam.

B. Results: Technical Details

Unconditional sampling. In Fig. 13, we provide more unconditional samples from our model from different viewpoints. All the samples have been produced by sampling $z \in \mathcal{N}(0, \mathbb{I})$ with a truncation trick [11] with the power $\psi = 0.7$. For the evaluation in the Table 1 in the main text, the implementation of the metrics MMD, JSD, COV from PointFlow [58] was used. Since FaceVerse contains samples grouped by subjects, the nearest neighbor of a ground truth scan is typically a scan of the same subject with a different expression. Because of that, we only select one ground truth sample per subject (with the same neutral expression for all subjects) to calculate COV. All FaceVerse scans are used to calculate MMD and JSD. As a distance measure between individual point clouds, aggregated over multiple observations in MMD and COV, we use Chamfer Distance (CD).

Ablating over the choice of the generative model architecture. The VAE used in our experiments is based on the Lightning Bolts library. The encoder follows the ResNet-18 architecture consisting of blocks of 2 convolutions each, with every second convolution with a stride of two (starting from the third) to downsample the activations spatially the increasing number of channels (64 in the first two blocks, then 128, 256, 256, 256, 512, 512 in the next blocks, respectively). The Lightning Bolts implementation adds two fully-connected layers on top of the encoder (one for the μ and one for the σ prediction). The dimension of the latent space equals 512. The decoder follows the architecture symmetric to the encoder, where the stride two for some

convolutions is replaced with a nearest-neighbor 2x upscaling.

For VQ-VAE implementation, we used the implementation of the VQ layer from vector-quantize-pytorch. Pixelcnn-pytorch served as a basis for the PixelCNN sampler implementation. Similarly to VAE, ResNet-18 encoder and decoder were used, with the exception that fewer downsampling operations have been used: they were introduced at each second layer (starting from the third), not each first layer. This is introduced to maintain a trade-off between the autoencoder quality and sampling ability, i.e. not to make PixelCNN operate in a too small latent space. The spatial resolution of the bottleneck is 32×32, which we found to be optimal, as the sampling performance of PixelCNN degrades from the top-left corner to the bottom-right corner and it is very noticeable already at the 64×64 bottleneck spatial resolution. The number of channels is 64, 128, 128, 32 for each two consecutive blocks, respectively. VQ-VAE is trained for 10K steps with batch size of 8, which we found to be enough to reach the sufficient visual quality of autoencoding. To facilitate the sampling, we obtain a dataset of VQ indices and learn PixelCNN to autoregressively sample from those for 200 epochs with a batch size of 32.

Behavior of the registration procedure. In Fig. 14, 15, 16, 17, we show how the mesh deforms as a result of the vector displacements optimization and normal displacements optimization.

Consistency of registrations. In Fig. 19, we demonstrate the analysis as to which template vertices are selected by the registration procedure to cover various regions of different meshes. Since we know the UV coordinates of all template vertices, this can be done by rendering the meshes with a UV checker texture image. Note that the long hair parts, such as pony tails, are mostly explained by the same regions of the layout as the vertices they originate from.

B.1. Applications

Fitting the latent code to a full scan. To fit the latent to the complete head scan, we have to apply preliminary steps, similar to the ones used to construct the training set. Firstly fit the FLAME to the scan, then apply our registration procedure to get a UV map U_{gt}. After that, we fit a $w \in \mathcal{W}+ \subset \mathbb{R}^{16\times512}$ latent code for the StyleGAN generator $g(w) : \mathcal{W}+ \to \mathbb{R}^{H\times W\times 3}$ to satisfy the following loss terms:

$$\begin{aligned}\mathcal{L}_{opt}^{full}(w|U_{gt}, \boldsymbol{\lambda}) \\ = \lambda^{\text{LPIPS}} \cdot \text{LPIPS}(g(w), U_{gt}) \\ + \lambda^{L_1} \cdot L_1(g(w), U_{gt}),\end{aligned}$$

where $L_1(\cdot,\cdot)$ is an average pixel-wise L1 distance between two images and $LPIPS(\cdot,\cdot)$ corresponds to the LPIPS score [62]. To calculate LPIPS, we cut the 256×256 UV maps (both predicted $U = g(w)$ and ground truth U_{gt} into sixteen 64×64 patches, evaluate LPIPS between the respective patches of U and U_{gt}, and average the obtained sixteen scores. The parameters of the loss equal to $\lambda^{\text{LPIPS}} = 0.1$ and $\lambda^{L_1} = 3$. The loss is being optimized via Adam algorithm with the learning rate of 10^{-2} for 1K steps. The w is initialized as the average latent predicted by the trained StyleGAN mapping network, evaluated over 10^5 codes $z \in \mathcal{N}(0, \mathbb{I})$.

Finally, we optimize for the StyleGAN noise (only for the scalp region of the UV space) to better fit the tiny details of the map U_{gt}. This step can be omitted in practice if fitting very high-frequency details is not required. Exactly the same loss terms are being optimized, this time not with respect to w but with respect to the StyleGAN noise tensors of all generator layers, while w remains fixed. The optimization is again carried out by Adam with the same learning rate and number of steps.

Fitting the latent code to a depth map. Fitting the latent representation to represent a partial observation poses a more challenging problem than trying to represent a full scan, since the resulting displacements must both resemble the original point cloud and complete it in a realistic way. This requires several changes to the fitting pipeline, described next.

Firstly, prior to applying the registration procedure to register part of the cloud P in the UV space, we identify the mask of points $m \in \{0, 1\}^{|V|}$ that are allowed to be offset by selecting only the points within the convex hull of the point cloud, expanded by 1.5x from its center to account for the possible important regions missing in the point cloud. The points below a certain horizontal plane are not accounted for when estimating the convex hull to disregard the shoulders and clothing, usually featured in NPHM raw scans. The level of the horizontal plane is selected as a 30% quantile of the coordinates of the points along the vertical axis. Masking out the points too far from the convex hull of the point cloud is especially important when the point cloud covers the minority of the geometry (e.g. if it is coming from a single depth map), since in this case, these points tend to pull in to cover the parts that the points inside the hull cannot explain (e.g. due to the regularizations), and this results in a non-plausible shape. For the registration procedure itself, stronger regularization parameters for the first stage have been selected, namely $\boldsymbol{\lambda}_{\text{Stage 1}} = (\lambda^{\text{Chamfer}}_{\text{Stage 1}}, \lambda^{\text{edge}}_{\text{Stage 1}}, \lambda^{\text{lapl}}_{\text{Stage 1}}) = (2{\cdot}10^3, 8{\cdot}10^5, 10^5)$. The correspondence pruning threshold, on the contrary, is raised to 10.0 for the first stage to allow the points to move farther while maintaining higher smoothness of the overall geometry due to stronger regularizations. For the second stage, the threshold is on the contrary reduced to 0.1 to penalize for large false movement of points along the template normals to explain the individual points of the cloud.

At the end of the registration, we refine the mask of the points by only selecting those of them that are sufficiently close to the fitted point cloud: $m_i^{\text{final}} = m_i \cdot [d(v_i + D_{\text{Stage 2},i}, nn(v_i + D_{\text{Stage 2},i}, P)) \leq t]$, where t defines the proximity threshold, and its optimal value depends on the sparsity of the cloud. For the point cloud formed from a dense depth map, we set $t = 0.1$, and for a sparse cloud with only 1% points of the original depth map left, we set $t = 0.3$. The regressed displacements and the mask are separately baked in the UV map as two independent images, 3-channel real-valued U and 1-channel binary M, respectively. In Fig. 18, we demonstrate the typical result of the partial registration stages.

Another important change lies in the latent fitting procedure. In our observations, the optimization of $w \in \mathcal{W}+$ latent code works great for the visible part but tends to produce displacements closer to the average shape for the non-visible part. We explain it by not strong enough supervision from the prior during fitting in $\mathcal{W}+$ space. To mitigate that effect, we first fit the $z \in \mathcal{Z} \subset \mathbb{R}^D$ latent code of the StyleGAN mapping network $map(z) : \mathcal{Z} \to \mathcal{W}+$, obtain the respective $w = map(z) \in \mathcal{W}+$ and regress the delta to the w code: Δw. We found that optimizing z code yields much better, yet rougher result of completion, and refining the map by regressing the Δw greatly improves fitting of the details.

In more detail, during the first z optimization step, we optimize the following loss:

$$\begin{aligned}&\mathcal{L}_{opt}^{z}(z|U_{gt}, \boldsymbol{\lambda}) \\ &\quad = \lambda^{\text{LPIPS}} \cdot \text{LPIPS}(g(map(z)) \cdot M, U_{gt} \cdot M) \\ &\quad + \lambda^{L_1} \cdot L_1(g(map(z)) \cdot M, U_{gt} \cdot M),\end{aligned}$$

Similarly to the $\mathcal{L}_{opt}^{full}$, we use $\lambda^{\text{LPIPS}} = 0.1$ and $\lambda^{L_1} = 3$. The z is initialized from $\mathcal{N}(0, \mathbb{I})$ and further optimized by Adam with the learning rate of 10^{-2} for 500 steps. Here and further, $\text{LPIPS}(\cdot, \cdot)$ and $L_1(\cdot, \cdot)$ follow the same expressions as for the full scan fitting.

During the second Δw optimization step, we optimize a similar expression with a few additional terms:

$$\begin{aligned}&\mathcal{L}_{opt}^{\Delta w}(\Delta w|z, U_{gt}, \boldsymbol{\lambda}) \\ &\quad = \lambda^{\text{LPIPS}} \cdot \text{LPIPS}(g(map(z) + \Delta w) \cdot M, U_{gt} \cdot M) \\ &\quad + \lambda^{L_1} \cdot L_1(g(map(z) + \Delta w) \cdot M, U_{gt} \cdot M) \\ &\quad + \lambda^{\text{LPIPS}}_{\text{preserve}} \cdot \text{LPIPS}(g(map(z) + \Delta w) \cdot (1 - M), \\ &\qquad\qquad g(map(z) \cdot (1 - M)) \\ &\quad + \lambda^{L_1}_{\text{preserve}} \cdot L_1(g(map(z) + \Delta w) \cdot (1 - M), \\ &\qquad\qquad g(map(z)) \cdot (1 - M)) \\ &\quad + \lambda^{\text{face}} \|g(map(z) + \Delta w) \cdot M^{\text{face}}\|_1,\end{aligned}$$

where M^{face} is a predefined mask of the face region in the UV space, reduced to the circle including the eyes, nose and mouth.

The third and second "preserve" terms are introduced to not let the map guided by the Δw optimization deviate much from the output corresponding to the regressed z in non-visible regions, which is essential due to the tendency of convergence to the average shape there when optimizing in the $\mathcal{W}+$ space. $\lambda^{\text{LPIPS}} = 0.1$ and $\lambda^{L_1} = 3$ remain the same as before, and $\lambda^{\text{LPIPS}}_{\text{preserve}} = 0.01$ and $\lambda^{L_1}_{\text{preserve}} = 0.3$ are selected $10\times$ less. The last regularization term is introduced to avoid hallucinations in the facial region that otherwise become visually apparent in the non-visible region of the map even in case of relatively small high-frequency inconsistency. λ^{face} is set to $\frac{10}{256}$. The optimization is carried out by Adam with the same learning rate of 10^{-2} for 500 steps. The Δw is initialized with zeros.

Finally, we optimize the StyleGAN noise to improve the details in the visible part. Despite that we consider this step optional, we found that it helps reconstruct more detail even for a sparse cloud. We optimize the same expression as $\mathcal{L}_{opt}^{z}$, with the difference that it is only being optimized w.r.t. the StyleGAN noise tensors (only in the scalp region). The only modification is the introduced regularization that equals to the sum of the noise tensors L2 norms. The optimization is carried out by Adam with the same learning rate of 10^{-2} for 500 steps. The coefficient of this regularization is equal to 10^{-5}.

In the Supplementary Video, we demonstrate more results of fitting the latent to the point clouds with different sparsity.

Animation. Here we expand on more details regarding applying displacements to a template, deforming over time. Compared to the simple unconditional scenario, where the displacements are also applied to a certain FLAME template, we have to introduce two key differences.

First, as mentioned in Subsec. A.1, to apply the displacements to the template, we apply Butterfly subdivision, the MeshLab implementation of which also smooths the surface. However, the result of Butterfly is not consistent over various FLAME templates and yields a bit different number of vertices every time. To solve that, we come up with *consistent subdivision*, i.e. the way to construct the same topology for every FLAME. To do that, we first apply Butterfly subdivision to an arbitrary scan, and for each vertex after the subdivision, we find which triangle of the original template it belongs to and the barycentric coordinates w.r.t. that triangle. Later, for every new template, the locations of the subdivided vertices are evaluated based on these triangles and barycentric coordinates. To handle the seam accurately, we consider each vertex of every triangle after subdivision individually, thus accounting for the duplicate vertices.

An artifact of such procedure is that the smoothness of

the surface, introduced in the MeshLab implementation of Butterfly subdivision, cannot be trivially transferred onto a new mesh this way. Because of this, the surface normals remain the same within the large triangles of the original template even after the subdivision, creating a non-appealing "tiling" effect. To mitigate that, we apply Laplacian smoothing [54] in its classical version to smooth the surface. In order to account for important subtle parts, we apply a different number of Laplacian smoothing iterations to different regions, namely, 3 times to the lips region, 5 times to the face skin (face except mouth, eyeballs and eye surroundings), and 10 times to the scalp and the neck. Since the realism of mouth, ears, and eyeballs is important for animation, they remain intact.

Second, as mentioned in the main text, we rotate the displacements according to the rotation of the surface normals of the template. To do that, we first estimate the local basis of the TBN space [2] for each FLAME in a sequence. This basis defines the normalized tangent $\boldsymbol{t_i^k}$, bitangent $\boldsymbol{b_i^k}$, and normal $\boldsymbol{n_i^k}$, pre-estimated for the i-th vertex of the FLAME template $F^k = FLAME(\text{shape}, \text{exp}^k, \text{jaw}^k, \text{headpose}^k)$. In addition, we estimate the TBN basis $(\boldsymbol{t_i^{\text{neutral}}}, \boldsymbol{b_i^{\text{neutral}}}, \boldsymbol{n_i^{\text{neutral}}})$ for a FLAME, corresponding to the same person and a neutral expression and pose $F^{\text{neutral}} = FLAME(\text{shape}, \mathbf{0}, \mathbf{0}, \mathbf{0})$. The displacements D, queried from the generated UV map U, are first transferred from the object space into the neutral TBN space:

$$D^{\text{TBN}} = \left((\boldsymbol{t_i^{\text{neutral}}} \cdot \boldsymbol{d_i}), (\boldsymbol{b_i^{\text{neutral}}} \cdot \boldsymbol{d_i}), (\boldsymbol{n_i^{\text{neutral}}} \cdot \boldsymbol{d_i})\right)_{i=1}^{|D|}$$

Then, for each of the sequence frames, we transfer them into object space, this time w.r.t. the TBN basis of the given frame:

$$D_k^{\text{object}} = \left(\begin{bmatrix} \boldsymbol{t_i^k} & \boldsymbol{b_i^k} & \boldsymbol{n_i^k} \end{bmatrix} \cdot \boldsymbol{d_i^{\text{TBN}}}\right)_{i=1}^{|D|}$$

(the $\boldsymbol{t_i^k}$, $\boldsymbol{b_i^k}$, $\boldsymbol{n_i^k}$, $\boldsymbol{d_i^{\text{TBN}}}$ vectors above treated as columns).

More examples of animations can be found in the Supplementary Video.

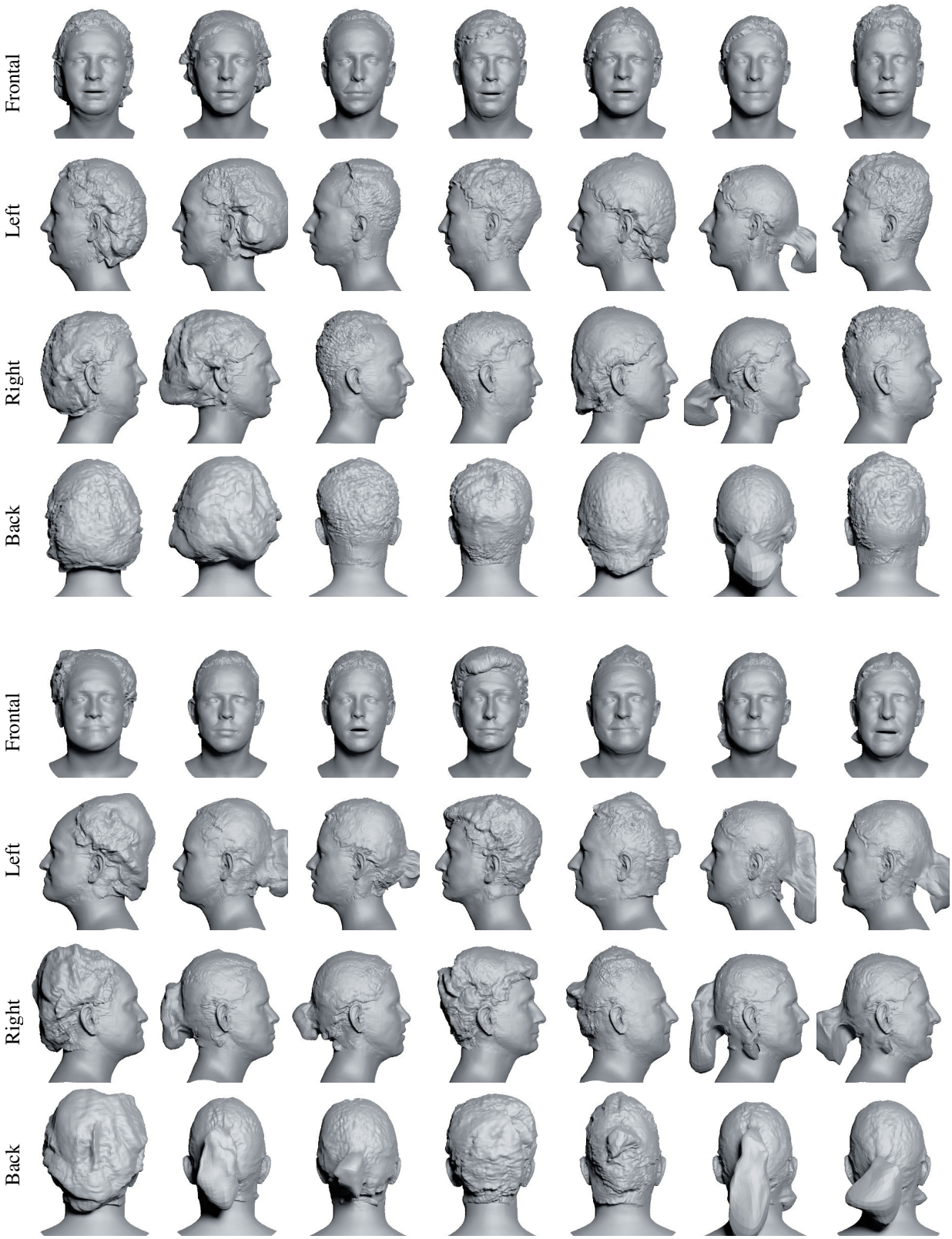

Figure 13. Additional results of the diversity and level of detail of the unconditionally sampled generations from HeadCraft. The generations are obtained by randomly sampling $z \sim \mathcal{N}(0, \mathbb{I})$ latent code of the generative model. The displacements, returned by the model, are applied to the random FLAMEs sampled from Gaussian distribution with statistics calculated over the NPHM dataset.

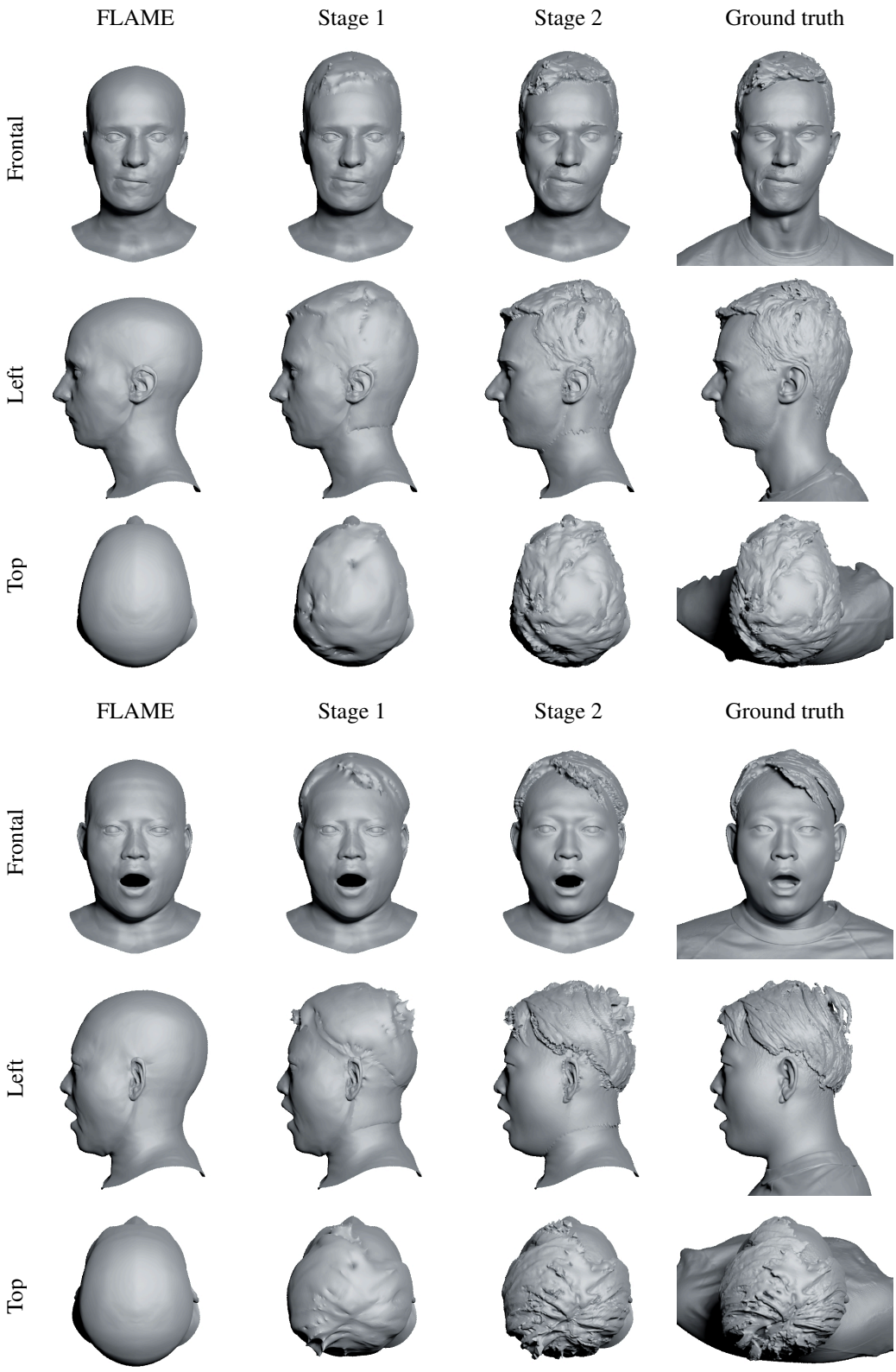

Figure 14. Additional demonstration of the two-stage registration. *Stage 1* corresponds to the vector displacements regression; *Stage 2* – to the refinement of the displacements along the normals. The second stage significantly improves the level of detail and allows us to match the high-frequency component of the scans, such as strands and subtle face features.

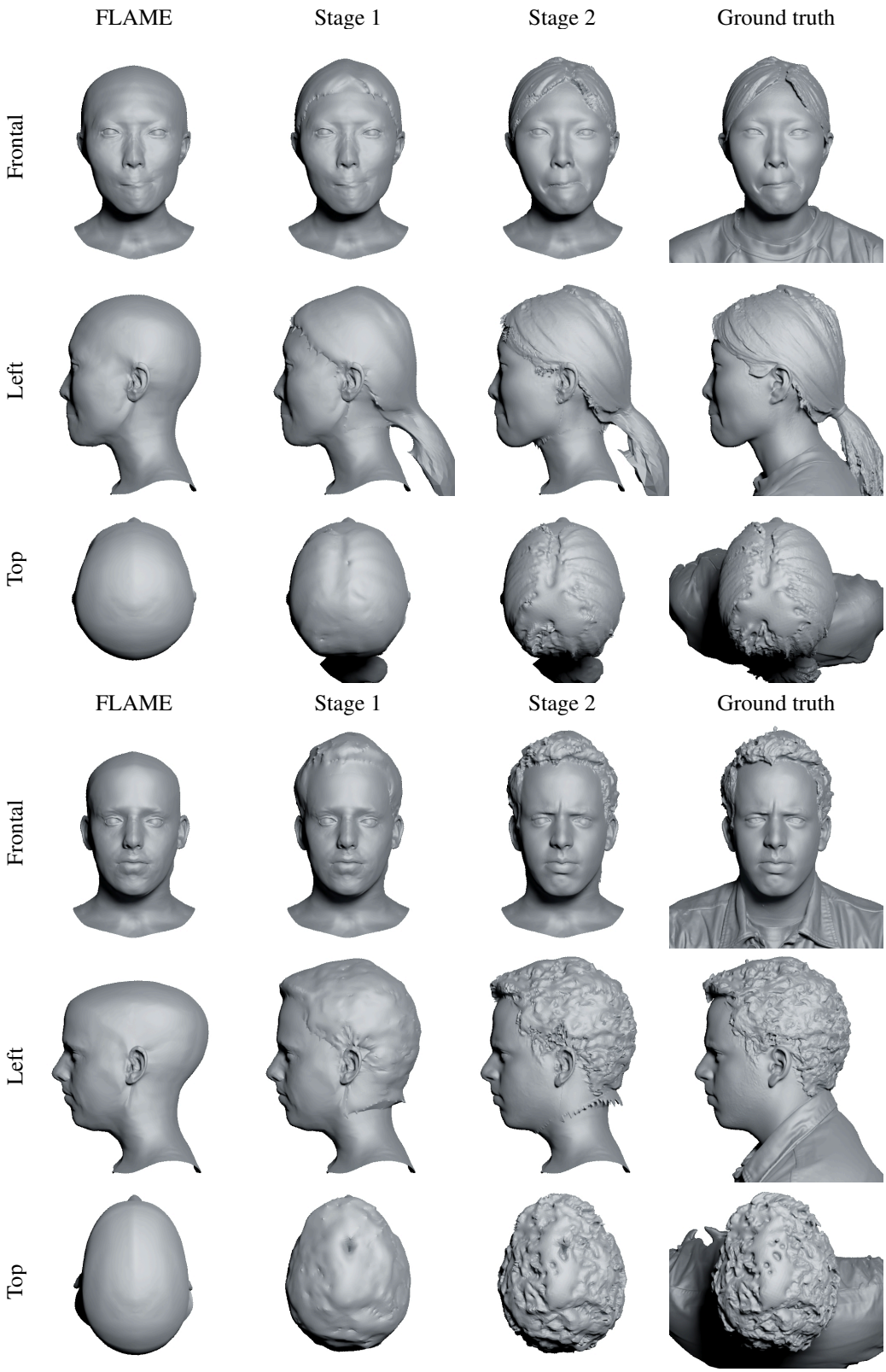

Figure 15. Additional demonstration of the two-stage registration. *Stage 1* corresponds to the vector displacements regression; *Stage 2* – to the refinement of the displacements along the normals. The second stage significantly improves the level of detail and allows us to match the high-frequency component of the scans, such as strands and subtle face features.

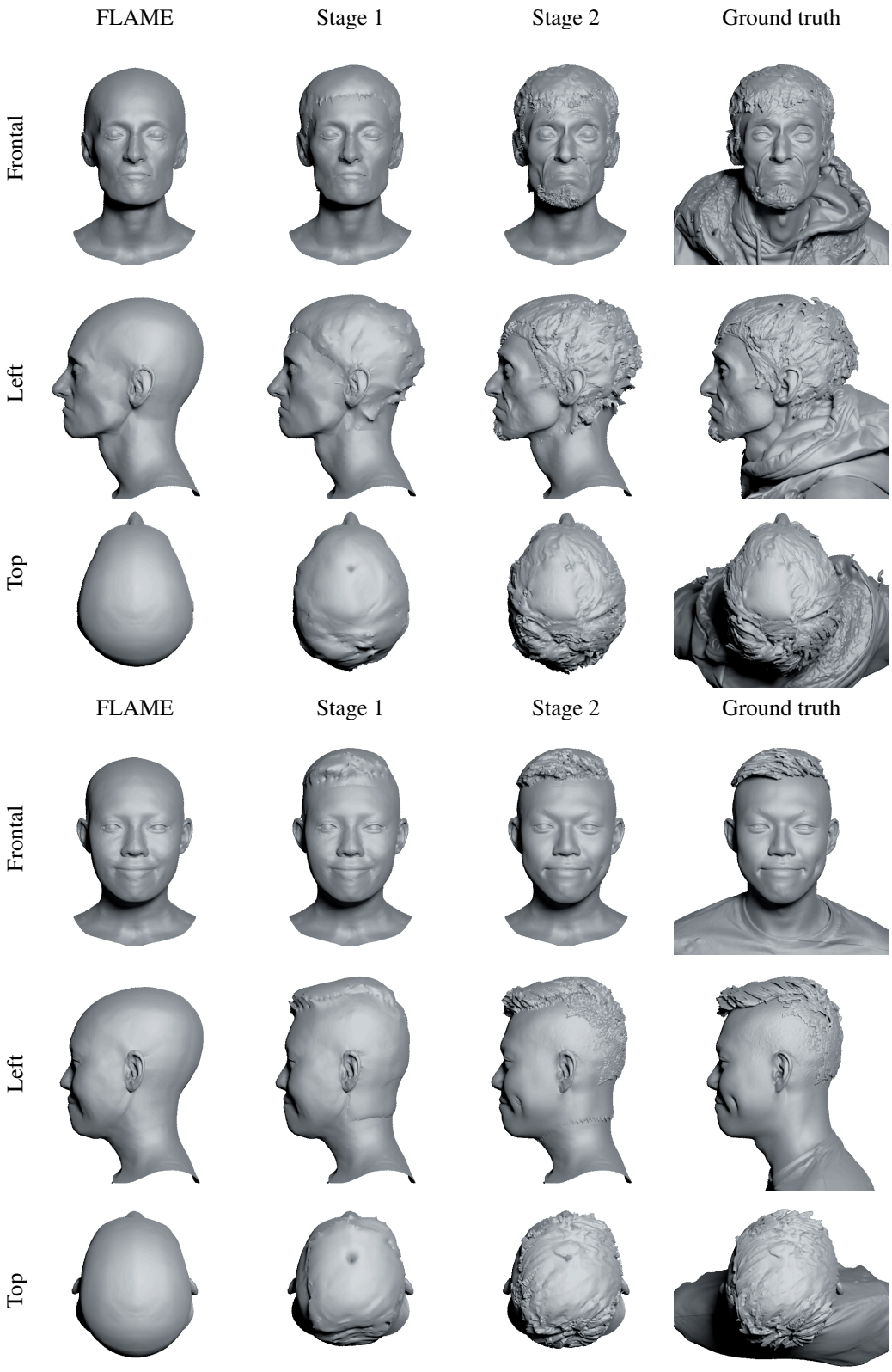

Figure 16. Additional demonstration of the two-stage registration. *Stage 1* corresponds to the vector displacements regression; *Stage 2* – to the refinement of the displacements along the normals. The second stage significantly improves the level of detail and allows us to match the high-frequency component of the scans, such as strands and subtle face features.

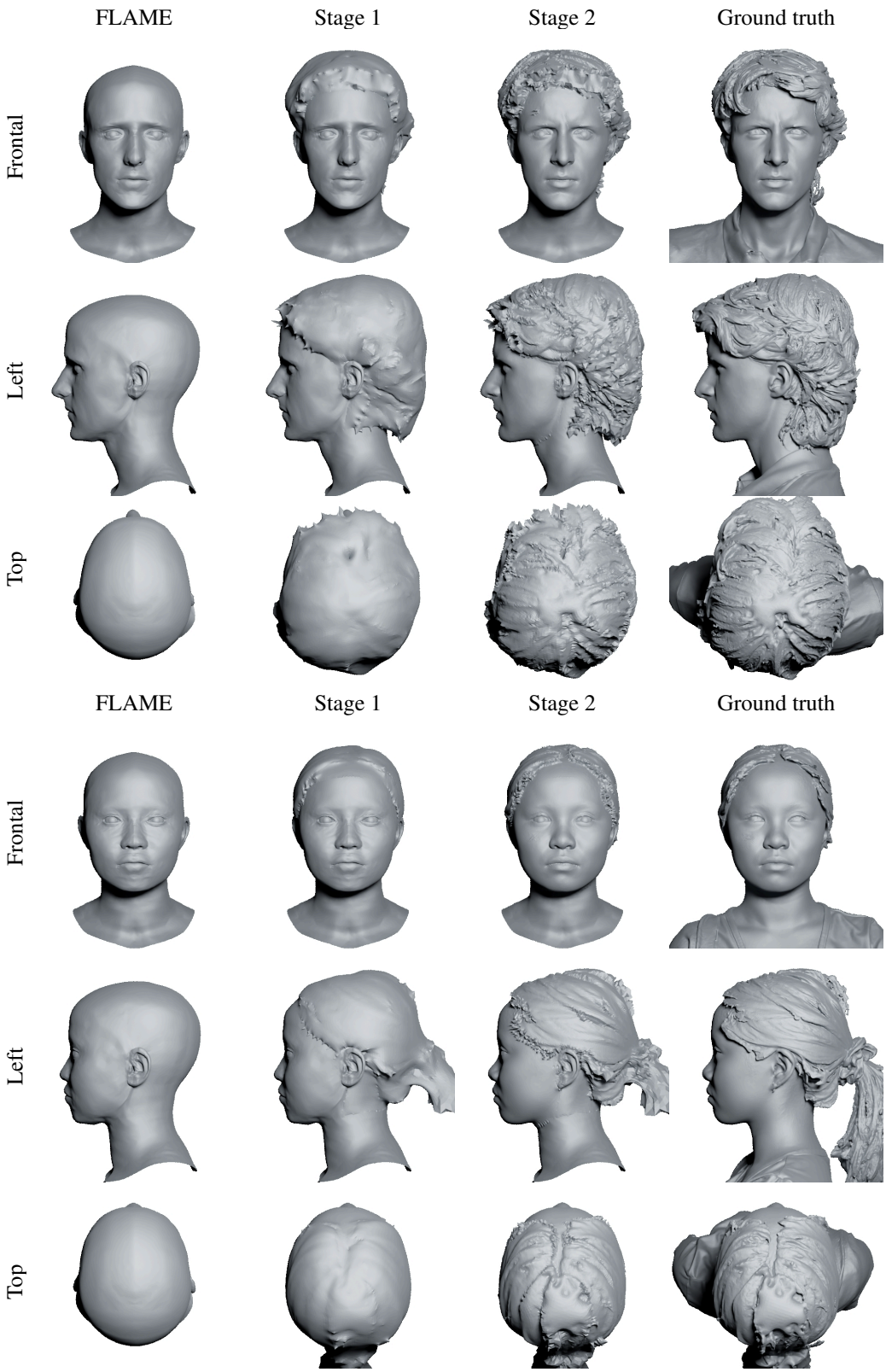

Figure 17. Additional demonstration of the two-stage registration. *Stage 1* corresponds to the vector displacements regression; *Stage 2* – to the refinement of the displacements along the normals. The second stage significantly improves the level of detail and allows us to match the high-frequency component of the scans, such as strands and subtle face features.

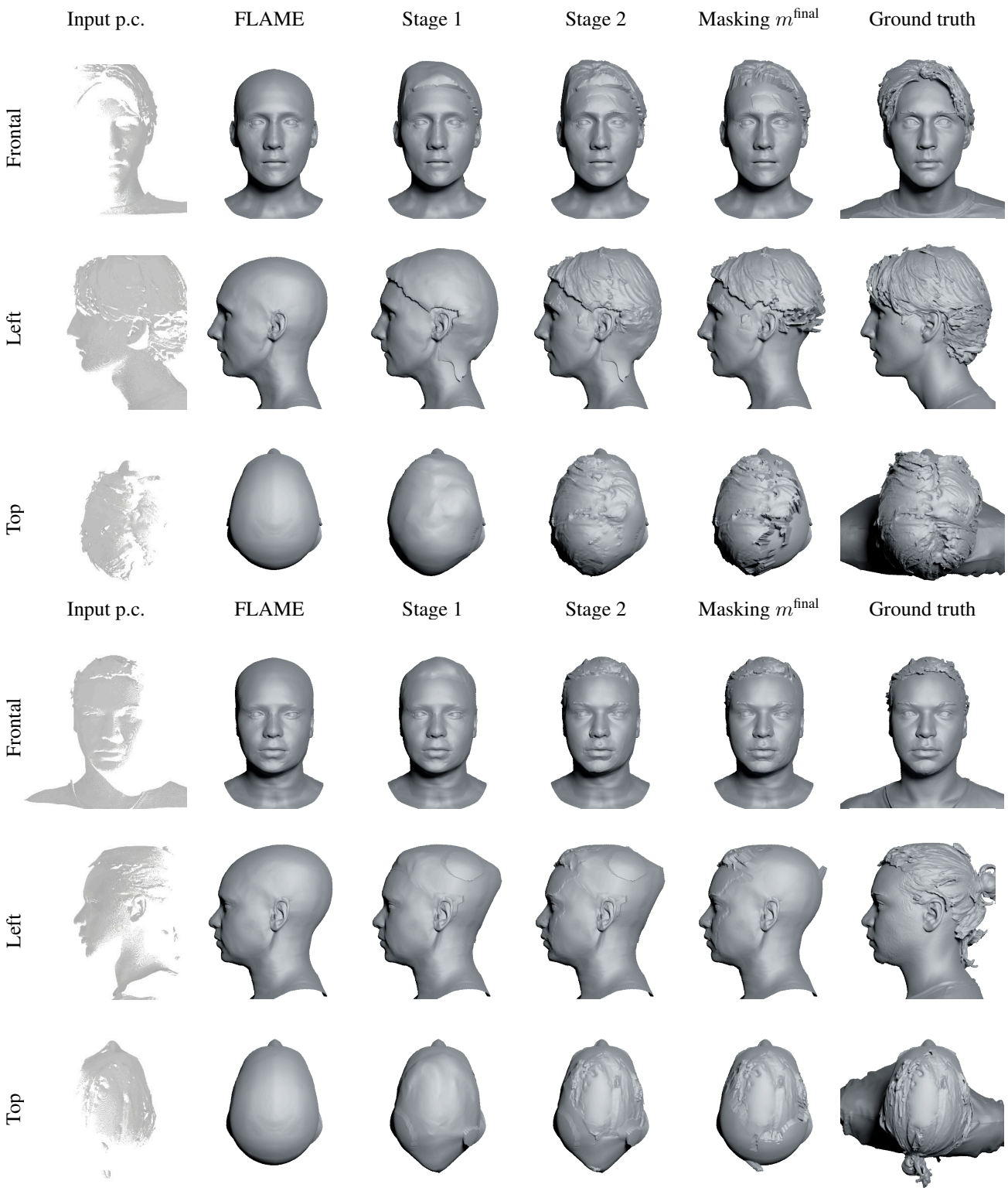

Figure 18. Demonstration of the stages of the partial registration procedure required to fit a part of the scan. The key difference between this procedure and the standard registration used to generate training data for HeadCraft is in the presence of only a part of the scan, e.g. a point cloud coming from the depth map. To overcome that obstacle, the displacements are being estimated only in the convex hull of the point cloud, and are subsequently filtered out by a separate mask m^{final}, leaving only the displacements close enough the ground truth scan (others are nullified in this visualization).

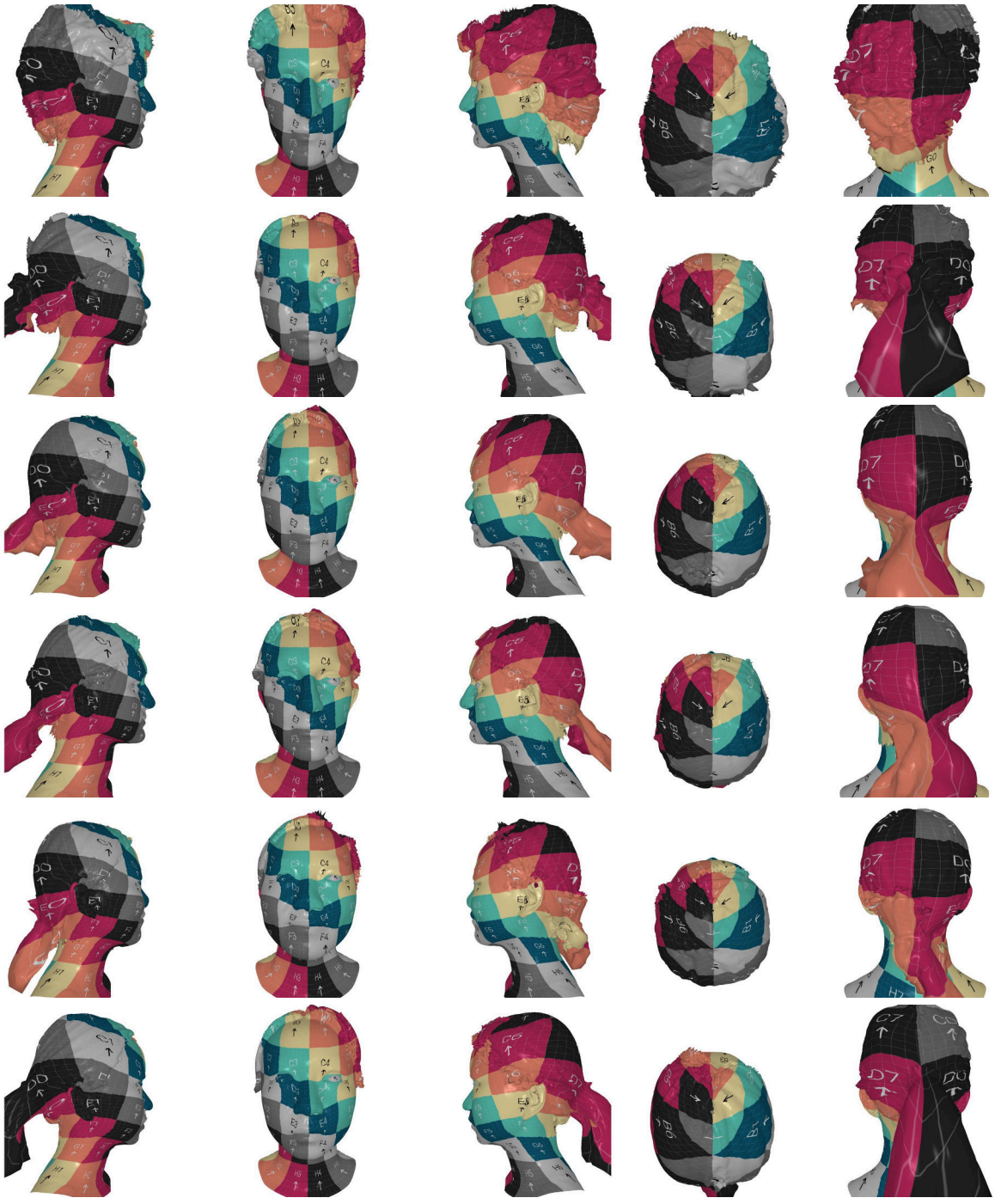

Figure 19. Consistency analysis of the registration. We demonstrate which template vertices are offset with the registration procedure to cover various regions of different meshes. Since we know the UV coordinates of all template vertices, this can be done by rendering the meshes with a *UV checker* texture image. For clarity of the visualization, the texture is applied to the standard FLAME layout. Note that the long hair parts, such as pony tails, are mostly explained by the same regions of the layout as the vertices they originate from.