NEO-3DF: Novel Editing-Oriented 3D Face Creation and Reconstruction

Peizhi Yan[1][0000−0002−6093−2964], James Gregson*, Qiang Tang[2], Rabab Ward[1], Zhan Xu[2], and Shan Du[3]**

[1] The University of British Columbia, Vancouver, BC, Canada
{yanpz,rababw}@ece.ubc.ca
[2] Huawei Technologies Canada, Burnaby, BC, Canada
[3] The University of British Columbia (Okanagan), Kelowna, BC, Canada
shan.du@ubc.ca

Abstract. Unlike 2D face images, obtaining a 3D face is not easy. Existing methods, therefore, create a 3D face from a 2D face image (3D face reconstruction). A user might wish to edit the reconstructed 3D face, but 3D face editing has seldom been studied. This paper presents such method and shows that reconstruction and editing can help each other. In the presented framework named NEO-3DF, the 3D face model we propose has independent sub-models corresponding to semantic face parts. It allows us to achieve both local intuitive editing and better 3D-to-2D alignment. Each face part in our model has a set of controllers designed to allow users to edit the corresponding features (e.g., nose height). In addition, we propose a differentiable module for blending the face parts and making it possible to automatically adjust the face parts (both the shapes and the locations) so that they are better aligned with the original 2D image. Experiments show that the results of NEO-3DF outperform existing methods in intuitive face editing and have better 3D-to-2D alignment accuracy (14% higher IoU) than global face model-based reconstruction. Code available at https://github.com/ubc-3d-vision-lab/NEO-3DF

Keywords: 3D Face Editing · 3D-to-2D Face Alignment.

1 Introduction

It has become popular for individuals recently to allow their faces to be used in 3D virtual reality (VR) applications and user-generated content games. For example, in a virtual conference or a virtual get-together meeting, the use of reconstructed 3D faces that resemble those of the real users helps the participants to identify the persons behind the avatars. Enabling a user to change some features, i.e. edit their reconstructed 3D face, can solve a main problem in existing 3D face reconstruction methods. As the reconstructed 3D face is usually

* This work was done when James Gregson was at Huawei Technologies Canada.
** Corresponding author. This work was supported by the University of British Columbia (Okanagan) [GR017752].

not accurate enough, the user may want to correct their reconstructed 3D face. Also when the user is not satisfied with some parts of their actual face, then they can modify those parts. While 3D face reconstruction has been well-studied in the past two decades [3,8,9,17,35,38], editing of 3D reconstructed faces has not. We believe there is a great potential for deep learning-based 3D face editing in many real-world applications, including 3D gaming, film-making, VR, and plastic surgery. 3D face editing can also benefit 2D face editing because the 3D face can provide explicit geometry information to 2D generative models [7,27,37].

3D editing is a challenging task. Editing a 3D shape usually requires strong spatial thinking ability, and traditional 3D editing tools are designed for use by artists. Moreover, editing of 3D faces is even more difficult than editing many other 3D shapes. This is because people are very sensitive to the appearance of a human face. Minor modifications of a 3D face could cause very different feelings in people. Because facial parts such as eyebrows and nose are relatively independent, segmenting the 3D face into parts and independently editing each part makes 3D face editing easier to handle [13, 14, 34]. In this work, we want to take one step further to bridge single-image 3D face reconstruction (SIFR) and 3D face editing. Since ambiguities such as depth and lighting will cause the reconstructed 3D face to be different from the actual face, the SIFR is an ill-posed problem. Recent works on SIFR display limited improvement in reconstruction when compared to the 3D ground-truth [8,11], whereas 3D-to-2D alignment, i.e. aligning the 3D face with the image it was reconstructed from, is still an open challenge [25]. Therefore, we also explore improving the 3D-to-2D alignment by taking advantage of 3D face editing. Inspired by [17, 38], we use 2D face semantic segmentation/parsing to provide ground truth in aligning the 3D face to the image. This requires us to segment our 3D face model based on the 2D face parsing, which is different from how a 3D face is segmented in [13, 14, 34].

In this paper, we propose a novel framework (which we named NEO-3DF) that allows the reconstructed 3D face to be edited locally. The features that can be edited are defined by the designer, but the extent of editing is determined by the user. Most SIFR methods use a global face model (i.e., the 3D face shape is modeled as a whole) which makes it extremely challenging for editing [8, 11, 21]. Our 3D face modeling however uses different sub-models for the different semantic segments (parts) of the face. This ensures that editing the shape of one part of a face does not cause changes in other parts, resulting in easier and more convenient editing. This approach also allows us to improve the 3D reconstruction accuracy using 2D supervision (see Section 3). We developed a way of using the As-Rigid-As-Possible (ARAP) method [33] to blend the face segments (parts), so the face appears natural at the segment boundaries. Importantly, we propose a differentiable version of ARAP that makes it possible to use gradient descent-based optimization to improve the alignment between the 3D reconstructed face segments and those of the original 2D image (i.e., 3D-to-2D alignment).

In summary, our main contributions are: (1) We proposed NEO-3DF, the first method that couples single-image 3D face reconstruction and intuitive 3D face editing. (2) The NEO-3DF is face-parts-based and follows 2D semantic

face segmentation, which makes it possible for an automatic shape adjusting process. (3) We proposed a differentiable version of ARAP to allow us to use 2D face segmentation as guidance to adjust the reconstructed 3D face shape automatically. (4) We associated editing controllers with the latent space and justified the effectiveness of its use in an automatic 3D-to-2D alignment process.

2 Related Works

3D Face Modeling and Reconstruction. The 3D face is usually represented by the 3D mesh structure, where the vertices and facets define the surface of the 3D face. A straightforward way to create a 3D face is by using a weighted average of a linear combination of a set of example 3D faces [36]. An improved version of this approach, the 3D Morphable Face Model (3DMM), transforms the scanned and processed example 3D faces to a vector space representation, using Principal Component Analysis (PCA) [3]. The PCA captures the primary modes of variation in the pre-collected 3D face dataset. Although 3DMM was proposed two decades ago, it is still widely used in current human face-related research and applications, such as 3D face reconstruction [24,38], controllable face image generation [7,37], and 3D face animation [2]. In recent years, research on the use of deep learning to learn the general 3D face model shows the advantage of neural network-based 3D face models over traditional 3DMMs [4,29,35].

Single-image 3D face reconstruction (SIFR) is an important application of both 3DMMs and neural network-based 3D face models [9]. The task is to reconstruct a 3D face from a given single-view face image. In most cases, the camera intrinsic and lighting conditions are unknown. In addition, for images where the faces appear identical, the actual 3D shape of these faces can be different due to potential ambiguities. Thus, the SIFR is an ill-posed problem, and we can only approximately reconstruct a 3D face from a single image. However, compared with accurate 3D reconstruction methods, using expensive 3D scanning devices or multi-view images, SIFR is more accessible to users of consumer electronics. We can roughly group the existing SIFR methods as fitting-based methods [3,38] and learning-based methods [8,12,17,35]. Generally, both fitting-based and learning-based methods use a 3D face model such as linear 3DMM [8,11,17,38] or non-linear neural network-based model [12,35] to serve as a priori knowledge. The main difference is that the fitting-based methods will iteratively optimize the variables to make the generated 3D face look more like the face in the photo. The learning-based method uses another parametric model (usually a neural network) to regress those variables.

General-Purpose 3D Mesh Editing. Many 3D mesh editing methods allow the user to modify the location of some vertices (constraint vertices) to edit the 3D mesh. The cage-based deformation (CBD) method warps the space enclosed by "cages" to deform the mesh surface [19]. CBD is usually used for large-scale deformations such as human body postures and is not ideal for 3D face editing because the latter focuses more on local details. Building the appropriate "cages"

is also a challenging task. The ARAP method is a detail-preserving mesh surface deformation method [33]. ARAP assumes that by making the local surface deformation close to rigid, one can preserve the local surface details. In editing 3D face shapes, neither CBD nor ARAP can prevent improper edits, this is because these methods do not have prior knowledge on 3D faces. In comparison, the blendshape methods solve the face shape editing problem by using a set of predefined shape offsets to deform a given face shape [22]. Although the concept of the blendshape is straightforward, developing intuitive meaningful blendshapes is labor-intensive. 3DMMs can be considered as a special type of blendshape method, where the blendshapes are derived from 3D scans and compressed by PCA. However, the eigenvectors do not have intuitive meaning.

3D Face Editing. In the video gaming industry, 3D face editing systems are usually designed by artists. For example, the artists can define meaningful controllers (e.g., nose height) with corresponding handcrafted blendshapes to allow users to easily change the shape of a 3D face [32]. Foti et al. proposed the Latent Disentanglement Variational Autoencoder (LD-VAE) and demonstrated that it can be used for constraint vertex-based face shape editing [13]. Because latent disentanglement in terms of the face parts is an objective during the training of LD-VAE, the local editing result is better than using a traditional 3DMM. Whereas this type of editing still requires the user to have some skills in arts. Ghafourzadeh et al. developed a face editing system that uses a face parts-based 3DMM (PB-3DMM) [14]. This method decomposes the face shape into five manually selected non-overlapping parts to gain local control. The models of these face parts are built using PCA on 135 3D face scans. They also use the method proposed by Allen et al. to find a linear mapping between anthropometric measurement space and the PCA coefficient spaces to achieve intuitive control/editing (such as changing the nose height) [1]. PB-3DMM is the first data-driven 3D face model that supports intuitive local editing. However, like traditional 3DMM [3], the PB-3DMM still relies on 3D scans, which are expensive to collect and process. Moreover, PB-3DMM uses a specially designed low-polygon mesh topology and is not optimized for single-image 3D face reconstruction and 3D-to-2D face alignment. We briefly summarize some important factors of PB-3DMM, LD-VAE and the proposed NEO-3DF in Table 1.

Table 1. Comparison of the NEO-3DF and other parts-based 3D face methods.

Method	Model	Mesh Info. (Vertices)	Face Parsing	Intuitive Editing	Parts-Blending	Back-prop.	SIFR
PB-3DMM [14]	multiple PCAs	specially designed (6,014)	✗	✓	✓	✗	✗
LD-VAE [13]	single VAE	FLAME [23] (71,928)	✗	✗	N/A	✓	✗
NEO-3DF (proposed)	multiple VAEs	modified BFM [8, 26] (35,709)	✓	✓	✓	✓	✓

3 The Proposed Method

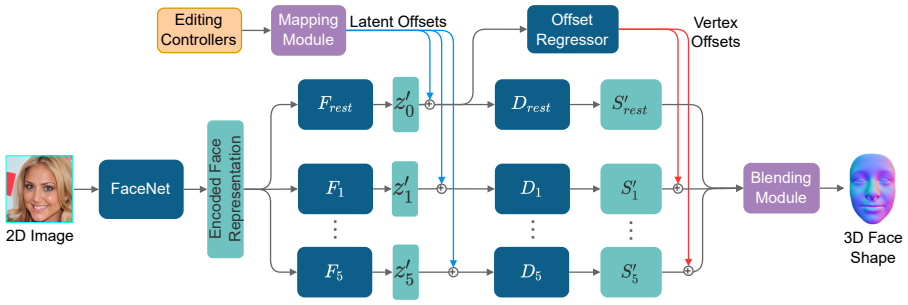

Fig. 1. Flowchart of the proposed NEO-3DF. FaceNet: The convolutional neural network encodes the 2D face image to a vector representation. Networks F_i estimate latent representations z'_i of corresponding face segments, and shape decoders D_i reconstruct the segment shape S'_i from z'_i. The Offset Regressor predicts offsets used to assemble the segment shapes to their correct location.

Overview. The proposed framework (see Fig. 1) supports single-view 3D face reconstruction and its further editing. In this work, we first reconstruct a neutral 3D face shape. The expression and texture of the 3D face can be added later through existing methods (such as [5] and [24]). We use a pre-trained FaceNet [31] as the face image encoder. In this work we segment our 3D face topology into six segments (parts). Each segment has its own editing controllers. These controllers determine the variables that can be changed in this segment (e.g., nose height). For each segment i, we have a regressor network F_i that estimates the latent encoding z'_i of the segment. Following each encoder F_i is the shape decoder D_i, which reconstructs the shape S'_i. The set of editing controllers enables intuitive face shape editing. To find the mapping between the editing controllers space and the neural network latent space, we use a similar way as used in [1,14]. The main idea is to construct a linear system representing the mapping function and solve the unknown parameters of the mapping function. The blending module uses ARAP optimization to derive an overall natural-looking face shape.

Semantic Face-Segment Based 3D Face Model. To develop a face model based on semantic face-segment, we use the Basel Face Model (BFM) [26] as the base model. We consider the face to have six segments: five semantic face segments (*eyebrows*, *eyes*, *nose*, *upper* and *lower lips*) and one segment for the *rest* of the face. Following [8], we remove the neck and ears from the original BFM mesh topology. The resulting 3D face mesh has 35,709 vertices. To automatically segment the 3D face, we use a similar approach as in [17]. The main idea is to render the randomly synthesized 3D faces to 2D, then run the 2D face parsing

model to get 2D parsing labels. Finally, we can map the 2D pixel-level labels back to 3D vertices. This automatic 3D face segmentation process ensures the 3D vertex labels are consistent with the mainstream face image semantic parsing, which is essential in our following work on fine-tuning the model, using ground-truth 2D face parsing masks as supervision. Fig. 2 (left) shows the face parts of our 3D face mesh topology.

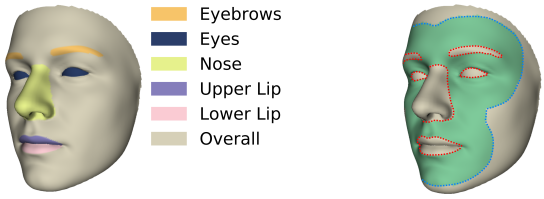

Fig. 2. Left: The six face segments considered. Right: Shape boundaries (semantic segment boundaries in red lines, outer boundary in blue line) and transition area (in green).

We denote the 3D shape S as a set of vertices $S = \{v_i | v_i \in \mathbb{R}^3\}_{i \in V}$, where V is the set of vertex indices of the shape S, v_i represents the 3D location of the i^{th} vertex in S. We use $S_k (k \in \{1..5\})$ to represent the five semantic face segments of eyebrows, eyes, nose, upper lip, and lower lip respectively. For convenience, we call S_ks the five-segments. The shape of the rest segment S_{rest} is a little different from S_ks, because it has a one-ring overlap with each of the S_ks. This one-ring overlap is important in our ARAP-based blending module which provides the curvature information at the segment boundaries (see Fig. 2).

We use Multi-Layer Perceptron (MLP) based Variational Autoencoders (VAEs) to learn the 3D shape representation. In total, we train six VAEs for each face segment i (E_i and D_i are encoder and decoder respectively). For S_{rest}, the latent representation is a vector $z_0 \in \mathbb{R}^{1\times 30}$, and for S_ks, $z_k \in \mathbb{R}^{1\times 10}$. We pre-process each segment shape in the dataset, except S_{rest}, to make the geometric center of the dataset's 3D bounding box as the origin of the 3D Cartesian coordinate system. The reason is that we expect the shape decoders of the five-segments to only care about the shape itself rather than the relative location of the segment on the face. The loss in training the VAEs is defined as:

$$\mathcal{L}_{VAE} = \lambda_{recon}\mathcal{L}_{recon} + \lambda_{KL}\mathcal{L}_{KL} + \lambda_{smooth}\mathcal{L}_{smooth}, \tag{1}$$

where $\mathcal{L}_{recon}$ is the shape reconstruction loss, $\mathcal{L}_{KL}$ is the Kullback-Leibler (KL) divergence loss, $\mathcal{L}_{smooth}$ is the cotangent-weight Laplacian loss, and the λs are loss term weights. Denote v and v' as the vertices of the ground-truth shape S and reconstructed shape S' respectively. We compute the reconstruction loss $\mathcal{L}_{recon}$ as $\sum_{i\in V} \|v_i - v'_i\|_2^2$. The $\mathcal{L}_{KL}$ is the KL divergence of the distribution of $z \in \mathbb{R}^{1\times d_z}$ from a multivariate normal distribution $\mathcal{N}(\mathbf{0}, \mathbf{I})$, where $\mathbf{I} \in \mathbb{R}^{d_z \times d_z}$

is the identity matrix and $\mathbf{0} \in \mathbb{R}^{1\times d_z}$ is a vector of zeros. The Laplacian loss $\mathcal{L}_{smooth}$ helps the decoder to generate smooth mesh surfaces.

Because the input to E_i is the 3D shape, we still need to couple the 2D face image encoding with the 3D shape decoders D_i. To do this, after VAE training we discard the VAE encoders E_i and add another set of regressor networks F_i between the 2D face image encoding and the 3D shape decoders D_i. To pre-train F_i, we freeze the parameters of the face image encoder and shape decoders. The loss function is the same as (1) except for discarding the $\mathcal{L}_{smooth}$ term. Here, $\mathcal{L}_{KL}$ acts as a regularization term to prevent over-fitting.

Assembling Face Segments and Network Fine-Tuning. There are two remaining issues. The first is that we re-centered the S_ks for training the corresponding shape decoders to disentangle the five-parts shapes from the global face shape. Therefore, these decoders are not aware of the relative position of S_ks to S_{rest}. The second issue is that the pre-trained FaceNet is not optimized for 3D face reconstruction.

To address the first issue, we train an offset regressor that takes S_{rest}'s latent representation z_0' as input to predict an offset $(0, \Delta \mathrm{y_k}, \Delta \mathrm{z_k})$ for each reconstructed segment shape S_k'. Because the human face is bilaterally symmetric, we do not add any offset along the x-axis. After adding the offsets, we can assemble the S_ks with S_{rest} to get a full 3D face $S'_{overall}$. Note that $S'_{overall}$ is likely to have visible segment boundaries without proper blending.

To address the second issue, we fine-tune all the networks except the shape decoders D_is to improve the reconstruction accuracy. Besides using the target shapes as supervision, we leverage the ground-truth 2D face parsing masks as additional supervision. The loss function for fine-tuning is defined as:

$$\mathcal{L} = \lambda_s \mathcal{L}_s + \lambda_p \mathcal{L}_p + \lambda_{KL} \mathcal{L}_{KL}. \tag{2}$$

$\mathcal{L}_s$ is the shape loss between $S_{overall}$ and $S'_{overall}$, the computation is the same as $\mathcal{L}_{recon}$. Define $\mathrm{P} \in \mathbb{R}^{\mathrm{H}\times\mathrm{W}\times 5}$ as the ground-truth parsing mask of S_ks, where H and W are the height and width of the mask. P_{ijk} represents the binary label that indicates whether a pixel at image location (i, j) belongs to the k^{th} ($k \in \{1..5\}$) segment or not. We include the estimated expression and pose and use a differentiable renderer to render $S'_{overall}$ to 2D, denoted by P'. The renderer is modified to use the vertex one-hot semantic labels as five-channel colors for shading. To prevent the gradient from vanishing when the displacement between P and P' is too large, we follow the idea in [17] to use Gaussian filter (we set the standard deviation to be 1 and filter size to be 10×10) to soften both P and P'. The parsing loss is then defined as: $\mathcal{L}_p = \|\mathrm{P} - \mathrm{P}'\|_2$. Here, the $\mathcal{L}_{KL}$ acts as a regularization term.

Face Segments Blending. An evident challenge for the segment-based face model is that it is not straightforward to ensure the transition between segments is natural-looking. Simply smoothing the region between different face segments

could destroy the high-frequency local structure. Our goal is to preserve the surface details as much as possible so that the final shape will still looks human and to make the transitions between face segments look natural without abrupt boundaries between segments. To achieve this, we use the As-Rigid-As-Possible (ARAP) [33] for mesh deformation optimization in our blending module.

Our blending module will only deform some areas on S'_{rest} that connects it to the five-segments. We call these areas the transition area (see Fig. 2 right). We define the transition area using the average BFM face shape. The transition area covers the vertices on S'_{rest} whose Euclidean distances are less than $30mm$ from their nearest vertices on the segment boundaries (see Fig. 2 right, the red lines). The outer contour of the transition area is the outer boundary (see Fig. 2 right, the blue lines). We use the ARAP method to deform the shape of the transition area. The constraints are formed by two groups of vertices – vertices from the five-segments lying on the segment boundaries and vertices from S'_{rest} lying on the outer boundary. In our experiments, we run the ARAP optimization for three iterations.

Intuitive Face Editing. We expect the editing to be intuitive. For example, one can increase/decrease the nose width by an adjustable strength. The overall idea is to find mappings between the pre-defined facial feature measurements space and the latent encoding space. Different from [14], our editing is not based on the exact measurement values (e.g., the exact width of the nose bridge). It is based on the direction (+ for increase, − for decrease) and strength (how many standard deviations from the mean shape). We refer to some studies on facial feature measurements [10, 16, 28, 30] and derive a list of features for editing purposes (see supplementary material). We grouped the features by the six segments. For example, nose height is a feature of the nose segment.

We denote N as the number of face shapes in the dataset and m as the number of features of the shape S (S can be the shape of any of the six segments). For each S in the dataset, we record the measurements of its features (the measurement is done automatically using pre-defined measurement rules) as a row in the matrix $X \in \mathbb{R}^{N\times m}$, and also record its latent encoding vector z as a row in the latent matrix $Z \in \mathbb{R}^{N\times d_z}$. We compute the average face shape $\bar{S}$ over the dataset and record its latent encoding as $\bar{z}$. Next, we subtract the measurements of average shape $\bar{S}$ from each row in X and denote the resultant matrix as ΔX. Similarly, we subtract $\bar{z}$ from each row in Z and denote the resultant matrix as ΔZ. Assume there is a linear mapping $M \in \mathbb{R}^{m\times d_z}$ such that $\Delta X M = \Delta Z$, we can estimate M by solving the linear system $M = \Delta X^{\dagger}\Delta Z$, where $\Delta X^{\dagger}$ is the Moore–Penrose pseudo inverse of ΔX.

We define the editing controllers as a vector $c \in \mathbb{R}^{1\times m}$. Each value in c controls a corresponding feature. We compute the standard deviation of each column in ΔX, and record it as $\Sigma \in \mathbb{R}^{1\times m}$. Finally, we represent the editing as the modification of latent encoding space: $\tilde{z} = z + \Delta z = z + (c \circ \Sigma)M$, where z is the latent encoding of the original shape S, and $\circ$ is the Hadamard product

operator. Using the shape decoder to decode the new latent encoding $\tilde{z}$, we get the edited shape. We need to compute an M and a Σ for each segment.

Differentiable Blending Module in Shape Adjusting. To further improve the 3D-to-2D alignment accuracy, we propose a novel differentiable version of ARAP in our blending module. Our differentiable ARAP is fully functional, transforming traditional ARAP optimization iterations into a sequence of differentiable operations through which loss function gradients can back-propagate. Our method is different from [38], which has proposed the use of the ARAP energy as a loss function but did not carry out the ARAP optimization itself. In contrast, our method back-propagates through the ARAP algorithm, including estimating neighborhood rotations and solving the linear system. In principle, our ARAP deformer module can be applied to many deep learning-based geometry processing tasks and used as a component in end-to-end training. However, here we use it to fine-tune the editing controllers due to the high vertex count of our facial model.

We define n_p as the number of vertices of the face mesh, and n_c as the number of constraint vertices. We represent the vertex positions of the 3D mesh as a matrix $P \in \mathbb{R}^{n_p \times 3}$, and the constraint vertex positions as a matrix $H \in \mathbb{R}^{n_c \times 3}$. We first build a sparse constraint matrix $C \in \mathbb{R}^{n_c \times n_p}$, such that $C_{ij} = 1$ only if the j^{th} vertex of the mesh is the i^{th} constraint vertex. Then we compute the combinatorial Laplacian of the 3D mesh denoted by $L \in \mathbb{R}^{n_p \times n_p}$. The reason for using the combinatorial Laplacian rather than the cotangent Laplacian is that the former only depends on the mesh topology so that we can reuse it on different 3D faces with the same mesh topology. We define $P' \in \mathbb{R}^{n_p \times 3}$ as the new vertex positions, R as the right-hand-side of linear system $AW = R$, where A is a sparse matrix (3) and $(W_{ij})_{i \in \{1..n_p\}, j \in \{1..3\}}$ is P'. The estimation of R needs both P and P', thus the optimization of P' is done by alternatively estimating R and solving the linear system $AW = R$ for multiple iterations. Because A only depends on the mesh topology, the system only needs to be formed and inverted once. Algorithm 1 is the pseudo-code of the proposed differentiable ARAP, where n_{iter} is the number of iterations we would like to run the ARAP optimization. In our experiment, we choose $n_{iter} = 3$. Please refer to the supplementary material for more details of our differentiable ARAP and the pseudo-code of $estimate\_rhs()$.

Unlike our fine-tuning of the network, we do not apply the Gaussian filter on both the ground-truth parsing P and the rendered parsing P$'$, instead we only apply distance-transform to P. We empirically find that this approach leads to a better result. Importantly, we directly adjust the editing controllers instead of adjusting the segment latent representations. In the optimization process, we only minimize $\|\mathrm{P} - \mathrm{P}'\|_2$. The optimization is performed on a single shape each time.

$$A = \begin{bmatrix} L^{\mathrm{T}}L & C^{\mathrm{T}} \\ C & \mathbf{0} \end{bmatrix} \in \mathbb{R}^{(n_p+n_c)\times(n_p+n_c)}. \tag{3}$$

Algorithm 1 Differentiable ARAP

Input: P, H, L, and A^{-1}
Output: P'
Require: $n_{iter} \geq 1$
$iter \leftarrow 0$
while $iter \leq n_{iter}$ **do**
  if $iter = 0$ **then**
    $R \leftarrow \begin{bmatrix} L^{\mathrm{T}}LP \\ H \end{bmatrix}$ //Initialize the right-hand-side.
  else
    $R \leftarrow \begin{bmatrix} estimate\_rhs() \\ H \end{bmatrix}$ //Estimate new right-hand-side.
  end if
  $W \leftarrow A^{-1}R$ //Solve the linear system $AW = R$.
  $P' \leftarrow (W_{ij})_{i \in \{1..n_p\}, j \in \{1..3\}}$ //Update new vertex positions.
  $iter \leftarrow iter + 1$
end while

4 Experiments

Data and Training. The face images we use in our experiments are from CelebAMask-HQ dataset (30,000 images) [20] and FFHQ dataset (70,000 images) [15]. We run Deng et al.'s method (Deep 3DMM) [8] on both datasets to get the estimated BFM coefficients for each image. An alternative option is to use the synthesis-by-analysis method [3] to estimate the coefficients, but it would be slow. Then, we use the BFM coefficients to recover the 3D faces and get 100,000 3D faces. Because the CelebAMask-HQ dataset comes with ground-truth semantic face parsing masks, we use its first 5,000 images to form the validation set. The rest of the 95,000 images and their 3D reconstructions are used to train the models. We use the 25,000 training images from the CelebAMask-HQ dataset and their corresponding semantic parsing masks to fine-tune our network. We also use all the 95,000 3D reconstructions in the training set to compute the mappings for intuitive editing. The loss term weights are set as follows: $\lambda_{recon} = 1$, $\lambda_{KL} = 0.01$, $\lambda_{smooth} = 1$, $\lambda_s = 1$, $\lambda_p = 1$. We use Adam optimizer with a constant learning rate of 10^{-4}. The training batch size is 16. Our framework has an average per-vertex mean squared error of 9.0×10^{-3} on validation data (3D mesh data is measured in millimeters).

3D-to-2D Segments Alignment. In the single-image 3D face reconstruction task, the best that can be done is to make the rendered 3D face look as close as possible to the face in the image. It is challenging for existing global-based 3D face reconstruction methods to make each 3D semantic face segment better align with its corresponding 2D image semantic segment. We presume that the global-based 3D face model has strong constraints to ensure the entire 3D face looks as natural as possible. The drawback is that small face segments such as the eyes and the lips may not look realistic and are prone to be misaligned. In

addition, the human visual system is very sensitive to tiny differences in a face. Thus, the global-based methods are usually not able to provide satisfactory reconstructions. Some research works [6, 18] try to mitigate this problem by using more realistic textures to cheat the human viewers. In this work, we take advantage of our segment-based face model to improve the shape accuracy by tuning each 3D semantic segment, so it better aligns with the image. Learning realistic texture will also benefit from better aligned 3D shapes [24]. We conjecture that improved registration of 3D geometry to semantic face segments will assist texture reconstruction models in learning realistic textures, although it is not a focus of the current work.

We use the average Intersection-over-Union (IoU) metric to evaluate the 3D-to-2D alignment accuracy. Table 2 compares the deep 3DMM (global method), PB-3DMM, LD-VAE, and the proposed NEO-3DF. Note that both PB-3DMM and LD-VAE are not designed for single-image 3D face reconstruction, and both methods originally use different mesh topologies other than BFM, which we use. To compare with PB-3DMM and LD-VAE, we use the BFM mesh topology and our parsing-based face segmentation. Because both PB-3DMM and LD-VAE are local models, we use the similar network architecture of NEO-3DF to give both methods the single-image 3D face reconstruction ability. For PB-3DMM, the shape decoders are from BFM. For LD-VAE, the shape decoder is pre-trained using the original LD-VAE code, but on our dataset. We fine-tuned both variants under the same setting as training NEO-3DF.

Table 2. 3D-to-2D alignment results on validation data. Metric: average IoU.

Part Name	**Deep 3DMM [8]**	**PB-3DMM [14]**	**LD-VAE [13]**	**NEO-3DF**
Eyebrows	0.312	0.302	0.279	**0.363**
Eyes	0.387	0.377	0.372	**0.553**
Nose	0.766	0.728	0.735	**0.789**
Upper Lip	0.483	0.482	0.436	**0.531**
Lower Lip	0.468	0.424	0.418	**0.614**
All Five Parts	0.593	0.566	0.560	**0.616**

For single-sample shape adjusting, we use Adam optimizer with a learning rate of 10^{-3}. We run 20 gradient-descent optimization iterations for each shape, and the shape adjusting takes around 31 seconds for each iteration (on a workstation with one Nvidia Tesla V100-PCIe GPU). On the validation data, the average IoU of five-segment shapes after fitting is 0.683 (discussed in Ablation Study), which is 11% higher than the raw network prediction (0.616 as shown in Table 2). Fig. 3 shows some visualization of the alignment performance comparison. For each method, we use one column to show the rendered 3D face on top of the original image, and another column to show the union minus the in-

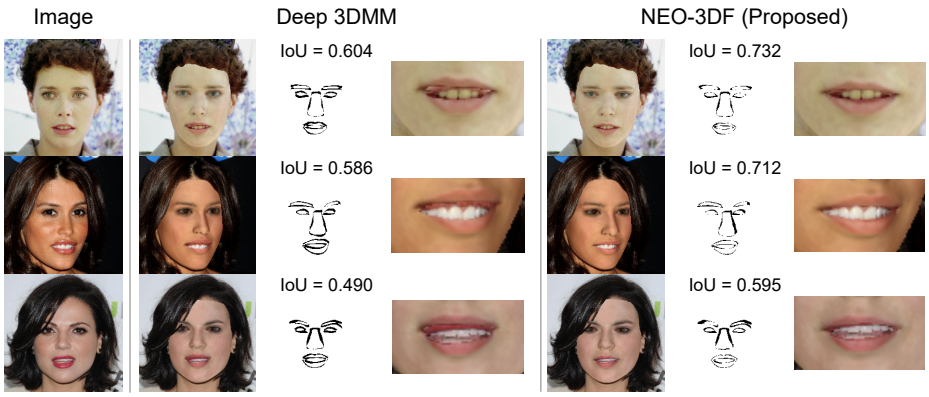

Fig. 3. 3D-to-2D semantic face segments alignment results. Compared with Deep 3DMM, we can see that our method gives better aligned facial segments (yields higher segmentation accuracy, as measured by IoU) and produces more consistent rendered results with original images.

tersection of P and P′. If the shapes of the face segments are better aligned to the 2D image, the black areas should be less visible.

Intuitive Editing. We compare our method with global VAE and LD-VAE [13]. We use the same MLP network structure for both global VAE and the LD-VAE, where the decoder structure is the same as non-linear 3DMM's shape decoder [35]. Our shape decoders have a similar decoder structure (same number of layers and hidden neurons), but each has a smaller output layer because the shape of a segment has fewer vertices than the entire face shape. We group the controllers by segments. In each group, we tune each controller separately to be -3σ and $+3\sigma$, then aggregate all the edited shapes and show the maximum change of each vertex location (Euclidean distance measured in millimeters) from the original mean face shape. The results are visualized as rendered 2D heatmaps (see Fig. 4). Fig. 5 shows the results of applying the same set of edits on the nose (*Tip* -3σ, *Breadth* $+3\sigma$, *Bridge Width* -3σ) using different methods. To demonstrate the application of our framework in customizing a reconstructed 3D face, we randomly apply a sequence of edits to the reconstructed faces. Fig. 6 shows some example intuitive editing results.

Ablation Study. We first investigate the necessity of employing the proposed offset regressor and blending module. Fig. 7 shows the original shape and the editing we made (lift the nose tip) using our unmodified framework, our framework without offset regressor, and our framework without blending module. Note that, in this experiment, only the nose is editable. Without the offset regressor, we need to use the absolute vertex locations to train the five-segments decoders. So the relative location of a five-part shape is still entangled into the latent rep-

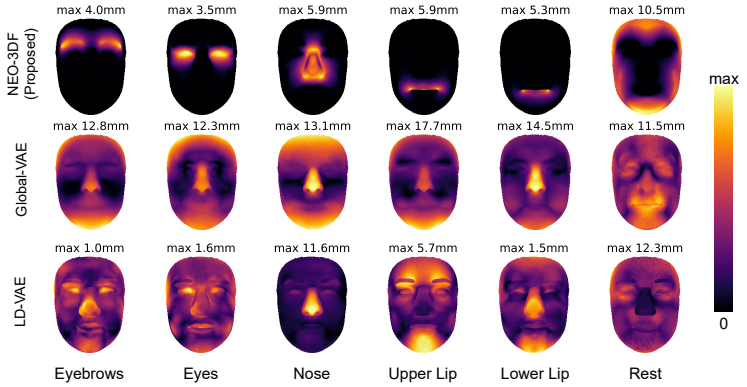

Fig. 4. Vertex location change heatmaps. For visualization purposes, the values in each heatmap are normalized to be from 0 to 1.0. The value above each heatmap indicates the highest value of that heatmap. We can see that our method gives near-perfect disentanglement while the other methods cannot isolate local edits well.

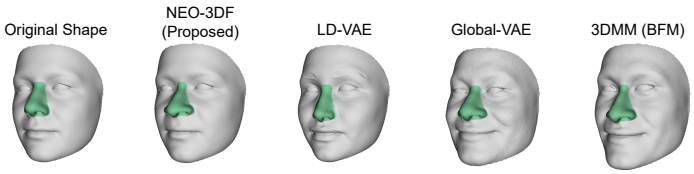

Fig. 5. Nose shape editing comparison.

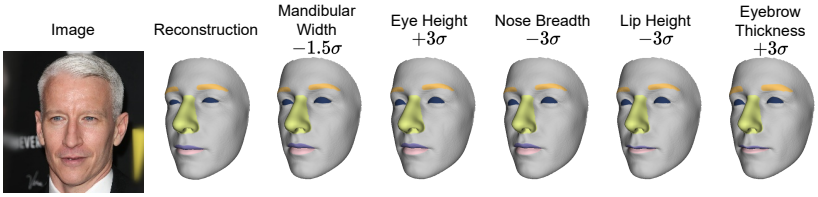

Fig. 6. Reconstruction and further editing examples. The edits are made cumulatively from left to right. Editing via our framework is more intuitive and expressive since subsequent changes (on different segments) will not affect earlier ones.

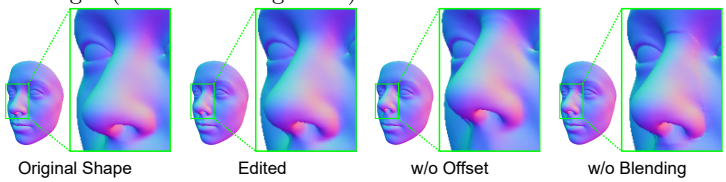

Fig. 7. Ablation study shows that the offset regressor and blending module are necessary for our framework. Without the offsets, alignment between segments is compromised, while visible seams appear around the edited segment without the blending module.

resentation. As a result, after removing the offset regressor, we see that the entire nose moves up when lifting the nose tip. The issue of eliminating the blending module is evident, as we can see the juncture around the nose.

We also investigate different ways of improving the 3D-to-2D alignment in our fitting-based single shape adjusting (see Table 3). We find that with the help of our differentiable ARAP, the best results are obtained by the fine-tuning of the editing controllers and the shape offsets. This yields 7% higher IoU than by directly adjusting the latent encodings and offsets (IoU is 0.632). Our explanation for this finding is that the editing controllers' space is less complex and more interpretable than the latent encoding space, thus reducing the learning complexity. Interestingly, we noticed that it does not work if we do not optimize the offsets and only optimize the editing controllers. We presume that the face segments must be roughly aligned to provide useful loss information.

Table 3. Single-sample automatic 3D part shapes adjustment results. (Metric: IoU averaged on validation data)

	w/o Blending	**w/ diff. ARAP Blending**
Latent Representations Only	0.621	0.626
Editing Controllers Only	0.581	0.566
Latent Representations & Offsets	0.629	0.632
Editing Controllers & Offsets	0.649	**0.683**

5 Conclusion

We proposed NEO-3DF, a 3D face creation framework that models the different segments of the 3D face independently. NEO-3DF is the first method that couples single-image 3D face reconstruction and 3D face intuitive editing. To train NEO-3DF, we created a 3D face dataset using existing large 2D face image datasets and an off-the-shelf 3D face reconstruction method named deep 3DMM. The face segment-based model of the 3D face structure makes face editing more intuitive and user-friendly. It also allows fine-tuning the reconstructed 3D face by using 2D semantic face segments as guidance and yields 14% improvement in IoU. We also proposed a differentiable version of ARAP to obtain an end-to-end trainable framework, which enables the automatic adjusting of the variables in the editing controllers. This resulted in an additional 5% improvement in IoU. The main limitation of our current framework is that the differentiable ARAP is computationally expensive and only used to fine-tune the result at the last construction stage. We will explore a less expensive method for blending the face segments in the future. We believe our concept of combining the 3D face editing and reconstruction also sets a new direction in 3D face reconstruction research.

References

1. Allen, B., Curless, B., Popović, Z.: The space of human body shapes: reconstruction and parameterization from range scans. ACM transactions on graphics (TOG) **22**(3), 587–594 (2003)
2. Bai, Z., Cui, Z., Liu, X., Tan, P.: Riggable 3d face reconstruction via in-network optimization. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 6216–6225 (2021)
3. Blanz, V., Vetter, T.: A morphable model for the synthesis of 3d faces. In: Proceedings of the 26th annual conference on Computer graphics and interactive techniques. pp. 187–194 (1999)
4. Bouritsas, G., Bokhnyak, S., Ploumpis, S., Bronstein, M., Zafeiriou, S.: Neural 3d morphable models: Spiral convolutional networks for 3d shape representation learning and generation. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 7213–7222 (2019)
5. Chang, F.J., Tran, A.T., Hassner, T., Masi, I., Nevatia, R., Medioni, G.: Expnet: Landmark-free, deep, 3d facial expressions. In: 2018 13th IEEE International Conference on Automatic Face & Gesture Recognition (FG 2018). pp. 122–129. IEEE (2018)
6. Chen, A., Chen, Z., Zhang, G., Mitchell, K., Yu, J.: Photo-realistic facial details synthesis from single image. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 9429–9439 (2019)
7. Deng, Y., Yang, J., Chen, D., Wen, F., Tong, X.: Disentangled and controllable face image generation via 3d imitative-contrastive learning. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 5154–5163 (2020)
8. Deng, Y., Yang, J., Xu, S., Chen, D., Jia, Y., Tong, X.: Accurate 3d face reconstruction with weakly-supervised learning: From single image to image set. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops. pp. 0–0 (2019)
9. Egger, B., Smith, W.A., Tewari, A., Wuhrer, S., Zollhoefer, M., Beeler, T., Bernard, F., Bolkart, T., Kortylewski, A., Romdhani, S., et al.: 3d morphable face models—past, present, and future. ACM Transactions on Graphics (TOG) **39**(5), 1–38 (2020)
10. Farkas, L.G., Kolar, J.C., Munro, I.R.: Geography of the nose: a morphometric study. Aesthetic plastic surgery **10**(1), 191–223 (1986)
11. Feng, Y., Feng, H., Black, M.J., Bolkart, T.: Learning an animatable detailed 3d face model from in-the-wild images. ACM Transactions on Graphics (ToG), Proc. SIGGRAPH **40**(4), 88:1–88:13 (Aug 2021)
12. Feng, Y., Wu, F., Shao, X., Wang, Y., Zhou, X.: Joint 3d face reconstruction and dense alignment with position map regression network. In: Proceedings of the European conference on computer vision (ECCV). pp. 534–551 (2018)
13. Foti, S., Koo, B., Stoyanov, D., Clarkson, M.J.: 3d shape variational autoencoder latent disentanglement via mini-batch feature swapping for bodies and faces. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 18730–18739 (2022)
14. Ghafourzadeh, D., Fallahdoust, S., Rahgoshay, C., Beauchamp, A., Aubame, A., Popa, T., Paquette, E.: Local control editing paradigms for part-based 3d face morphable models. Computer Animation and Virtual Worlds **32**(6), e2028 (2021)

15. Karras, T., Laine, S., Aittala, M., Hellsten, J., Lehtinen, J., Aila, T.: Analyzing and improving the image quality of stylegan. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 8110–8119 (2020)
16. Kesterke, M.J., Raffensperger, Z.D., Heike, C.L., Cunningham, M.L., Hecht, J.T., Kau, C.H., Nidey, N.L., Moreno, L.M., Wehby, G.L., Marazita, M.L., et al.: Using the 3d facial norms database to investigate craniofacial sexual dimorphism in healthy children, adolescents, and adults. Biology of sex differences **7**(1), 1–14 (2016)
17. Koizumi, T., Smith, W.A.: Shape from semantic segmentation via the geometric rényi divergence. In: Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision. pp. 2312–2321 (2021)
18. Lattas, A., Moschoglou, S., Gecer, B., Ploumpis, S., Triantafyllou, V., Ghosh, A., Zafeiriou, S.: Avatarme: Realistically renderable 3d facial reconstruction" in-the-wild". In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 760–769 (2020)
19. Le, B.H., Deng, Z.: Interactive cage generation for mesh deformation. In: Proceedings of the 21st ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games. pp. 1–9 (2017)
20. Lee, C.H., Liu, Z., Wu, L., Luo, P.: Maskgan: Towards diverse and interactive facial image manipulation. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2020)
21. Lee, G.H., Lee, S.W.: Uncertainty-aware mesh decoder for high fidelity 3d face reconstruction. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 6100–6109 (2020)
22. Lewis, J.P., Anjyo, K., Rhee, T., Zhang, M., Pighin, F.H., Deng, Z.: Practice and theory of blendshape facial models. Eurographics (State of the Art Reports) **1**(8), 2 (2014)
23. Li, T., Bolkart, T., Black, M.J., Li, H., Romero, J.: Learning a model of facial shape and expression from 4d scans. ACM Transactions on Graphics (TOG) **36**(6), 1–17 (2017)
24. Lin, J., Yuan, Y., Shao, T., Zhou, K.: Towards high-fidelity 3d face reconstruction from in-the-wild images using graph convolutional networks. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 5891–5900 (2020)
25. Martyniuk, T., Kupyn, O., Kurlyak, Y., Krashenyi, I., Matas, J., Sharmanska, V.: Dad-3dheads: A large-scale dense, accurate and diverse dataset for 3d head alignment from a single image. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 20942–20952 (2022)
26. Paysan, P., Knothe, R., Amberg, B., Romdhani, S., Vetter, T.: A 3d face model for pose and illumination invariant face recognition. In: 2009 sixth IEEE international conference on advanced video and signal based surveillance. pp. 296–301. Ieee (2009)
27. Piao, J., Sun, K., Wang, Q., Lin, K.Y., Li, H.: Inverting generative adversarial renderer for face reconstruction. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 15619–15628 (2021)
28. Ramanathan, N., Chellappa, R.: Modeling age progression in young faces. In: 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06). vol. 1, pp. 387–394. IEEE (2006)
29. Ranjan, A., Bolkart, T., Sanyal, S., Black, M.J.: Generating 3d faces using convolutional mesh autoencoders. In: Proceedings of the European Conference on Computer Vision (ECCV). pp. 704–720 (2018)

30. Rhee, S.C., Woo, K.S., Kwon, B.: Biometric study of eyelid shape and dimensions of different races with references to beauty. Aesthetic plastic surgery **36**(5), 1236–1245 (2012)
31. Schroff, F., Kalenichenko, D., Philbin, J.: Facenet: A unified embedding for face recognition and clustering. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 815–823 (2015)
32. Shi, T., Yuan, Y., Fan, C., Zou, Z., Shi, Z., Liu, Y.: Face-to-parameter translation for game character auto-creation. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 161–170 (2019)
33. Sorkine, O., Alexa, M.: As-rigid-as-possible surface modeling. In: Symposium on Geometry processing. vol. 4, pp. 109–116 (2007)
34. Tena, J.R., De la Torre, F., Matthews, I.: Interactive region-based linear 3d face models. In: ACM SIGGRAPH 2011 papers, pp. 1–10. ACM (2011)
35. Tran, L., Liu, X.: Nonlinear 3d face morphable model. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 7346–7355 (2018)
36. Vetter, T., Blanz, V.: Estimating coloured 3d face models from single images: An example based approach. In: European conference on computer vision. pp. 499–513. Springer (1998)
37. Wood, E., Baltrusaitis, T., Hewitt, C., Dziadzio, S., Cashman, T.J., Shotton, J.: Fake it till you make it: Face analysis in the wild using synthetic data alone. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 3681–3691 (2021)
38. Zhu, W., Wu, H., Chen, Z., Vesdapunt, N., Wang, B.: Reda: reinforced differentiable attribute for 3d face reconstruction. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 4958–4967 (2020)