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

Peizhi Yan, James Gregson, Qiang Tang, Rabab Ward, Zhan Xu and Shan Du

Paper ID 111

We provide more technical details as well as results in this supplementary material.

1 Neural Network Architectures

The face image encoder in our framework is a FaceNet [7]. We use a Pytorch implementation of the FaceNet [2]. The network architecture is Inception-v1 [9], and it was trained on the VGGFace2 dataset [1]. The input shape of the FaceNet is $224 \times 224 \times 3$ (RGB color channel). The encoded face representation vector (output of the FaceNet) is 512-dimensional.

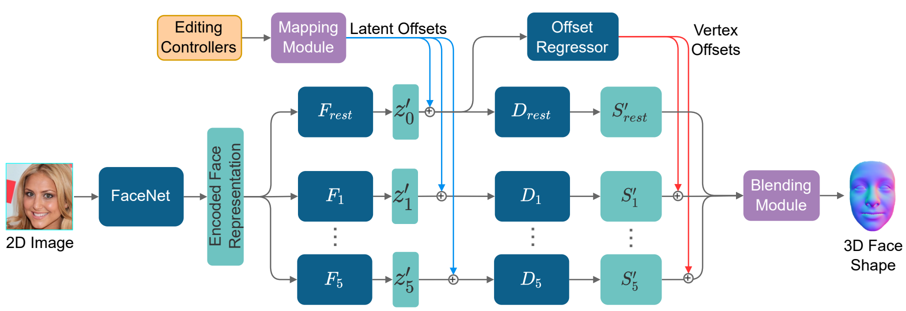

Fig. 1. The proposed NEO-3DF framework for single-image 3D face reconstruction and editing.

Except for FaceNet, in our framework, other neural networks (E_i, F_i, D_i, and the offset regressor network) are Multilayer Perceptron (MLP) networks. E_i and D_i constitute an VAE network that learns to reconstruct 3D shape S_i. The shape latent regressors F_i are added to replace the E_i to couple the FaceNet encoder with the shape decoders D_i in single-image 3D face reconstruction task (see Fig. 1). Table 1 to Table 4 show the description of network architectures of E_i, F_i, D_i, and the offset regressor network respectively. $|V_i|$ is the number of vertices of segment i, and d_{z_i} is the dimension of latent representation of segment i. Note that, the output of offset regressor network has 10 dimensions because only five segments need offset along y-axis (up and down) and z-axis (forward and backward).

Table 1. Shape Encoder Network (E_i) Architecture

Layer	Previous Layer	Activation	Input Size	Output Size
input	N/A	N/A	N/A	$[batch\_size, \|V_i\| \times 3]$
fc-1	input	ReLU	$[batch\_size, \|V_i\| \times 3]$	$[batch\_size, 128]$
fc-2	fc-1	ReLU	$[batch\_size, 128]$	$[batch\_size, 64]$
fc-mu	fc-2	Linear	$[batch\_size, 64]$	$[batch\_size, d_{z_i}]$
fc-sigma	fc-2	Exponential	$[batch\_size, 64]$	$[batch\_size, d_{z_i}]$

Table 2. Latent Regressor Network (F_i) Architecture

Layer	Previous Layer	Activation	Input Size	Output Size
input	N/A	N/A	N/A	$[batch\_size, 512]$
fc-1	input	ReLU	$[batch\_size, 512]$	$[batch\_size, 128]$
fc-2	fc-1	ReLU	$[batch\_size, 128]$	$[batch\_size, 64]$
fc-mu	fc-2	Linear	$[batch\_size, 64]$	$[batch\_size, d_{z_i}]$
fc-sigma	fc-2	Exponential	$[batch\_size, 64]$	$[batch\_size, d_{z_i}]$

Table 3. Shape Decoder Network (D_i) Architecture

Layer	Previous Layer	Activation	Input Size	Output Size
input	N/A	N/A	N/A	$[batch\_size, d_{z_i}]$
fc-1	input	ReLU	$[batch\_size, d_{z_i}]$	$[batch\_size, 128]$
fc-2	fc-1	ReLU	$[batch\_size, 128]$	$[batch\_size, 512]$
fc-output	fc-2	Linear	$[batch\_size, 512]$	$[batch\_size, \|V_i\| \times 3]$

Table 4. Offset Regressor Network Architecture

Layer	Previous Layer	Activation	Input Size	Output Size
input	N/A	N/A	N/A	$[batch\_size, d_{z_0}]$
fc-1	input	ReLU	$[batch\_size, d_{z_0}]$	$[batch\_size, 32]$
fc-2	fc-1	ReLU	$[batch\_size, 32]$	$[batch\_size, 32]$
fc-output	fc-2	Linear	$[batch\_size, 32]$	$[batch\_size, 10]$

2 Differentiable As-Rigid-As-Possible

We briefly review the mathematical notations used in the proposed differentiable ARAP in Table 5. The pseudo-code for our differentiable ARAP is in Algorithm 1, and the flowchart of it is in Fig. 2. Note that L is the combinatorial Laplacian ($L = Deg - Adj$, where Deg and Adj are the degree matrix and the adjacency matrix of the mesh graph topology). Although using cotangent-weight Laplacian (as used in the original ARAP [8]) will give better results, the computational overhead is much higher than using simple combinatorial Laplacian. Therefore, we only use the cotangent-weight Laplacian to derive the final result while using combinatorial Laplacian in fine-tuning. In Algorithm 1, A^{-1} is pre-computed for a given mesh. We treat L and A^{-1} as constant matrices, thus not backpropagating the error through it.

Table 5. Mathematical Notations for Differentiable ARAP

Notation	Space	Description
n_p	$\mathbb{R}$	The number of vertices of the face mesh.
n_c	$\mathbb{R}$	The number of constraint vertices.
P	$\mathbb{R}^{n_p \times 3}$	The vertex positions (coordinates).
H	$\mathbb{R}^{n_c \times 3}$	The constraint vertex positions.
C	$\mathbb{R}^{n_c \times n_p}$	The sparse constraint matrix, such that $C_{ij} = 1$ only if the j^{th} vertex of the mesh is the i^{th} constraint vertex.
L	$\mathbb{R}^{n_p \times n_p}$	The Laplacian matrix of the face mesh.
P'	$\mathbb{R}^{n_p \times 3}$	The new vertex positions of the face mesh.
A	$\mathbb{R}^{(n_p+n_c)\times(n_p+n_c)}$	$A = \begin{bmatrix} L^{\mathrm{T}}L & C^{\mathrm{T}} \\ C & \mathbf{0} \end{bmatrix}$
W	$\mathbb{R}^{(n_p+n_c)\times 3}$	The result matrix of linear system $AW = R$, and the first n_p rows $(W_{ij})_{i\in\{1..n_p\},j\in\{1..3\}}$ is P'.
R	$\mathbb{R}^{(n_p+n_c)\times 3}$	The right-hand-side matrix of linear system $AW = R$.

Except for the initial iteration (when *iter* is 0), we need to estimate the new right-hand-side R using $estimate\_rhs()$. Here, we take advantage of the fact that L is a sparse matrix to reduce computational overhead. We define $N(i)$ as the set of neighbor vertex indices of the i^{th} vertex, and $N(i)_k$ as the k^{th} neighbor vertex index (neighbor vertices arranged in index ascending order) of the i^{th} vertex. The pseudo-code for $estimate\_rhs()$ is in Algorithm 2. Note that the pseudo-code only conveys the idea of our algorithm. It is not optimized for a particular programming language. When implementing it, consider using matrix operations to reduce the explicit loops.

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

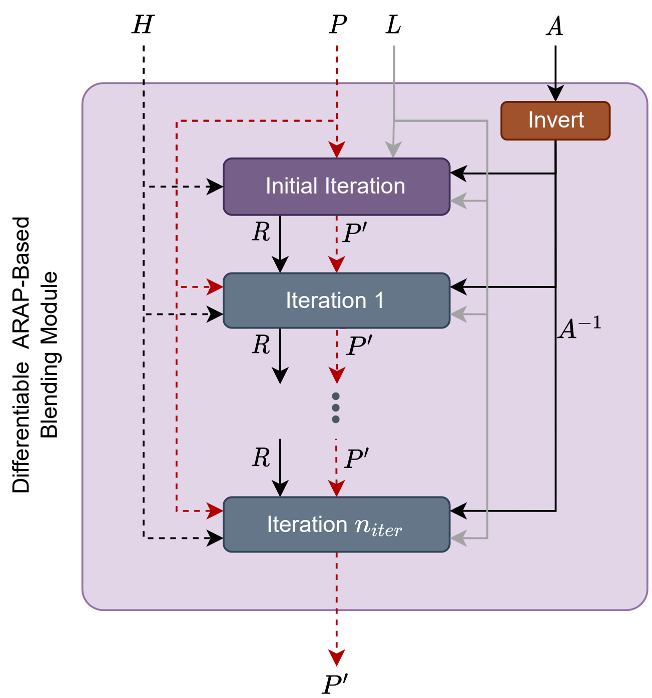

Fig. 2. Flowchart of our differentiable ARAP-based blending module. Dashed lines indicate that the loss can backpropagate through.

Algorithm 2 Estimate Right-Hand-Side

Input: P, P', L
Output: $estimate\_rhs()$
Prepare Γ according to the mesh topology.
// Step 1: Get the un-deformed and deformed 1-ring vertex positions relative to each vertex, weighted by the Laplace edge weight.
$\Omega \leftarrow \mathbf{0}^{n_p \times n_p \times 3}$ // $\Omega \in \mathbb{R}^{n_p \times n_p \times 3}$ is a sparse matrix.
$\Omega' \leftarrow \mathbf{0}^{n_p \times n_p \times 3}$ // $\Omega' \in \mathbb{R}^{n_p \times n_p \times 3}$ is a sparse matrix.
for $i \in \{1..n_p\}$ **do**
    for $j \in N(i)$ **do**
        $(\Omega_{ijm})_{m \in \{1..3\}} \leftarrow (P_{im} - P_{jm}) \cdot L_{ij}$
        $(\Omega'_{ijm})_{m \in \{1..3\}} \leftarrow (P'_{im} - P'_{jm}) \cdot L_{ij}$
    end for
end for
// Step 2: Estimate the rotation from the un-deformed to the deformed mesh using the singular value decomposition (SVD).
$\mathbf{R} \leftarrow \mathbf{0}^{n_p \times 3 \times 3}$ // Initialize the rotations.
for $i \in \{1..n_p\}$ **do**
    $B \leftarrow \mathbf{0}^{|N(i)| \times 3}$
    $B' \leftarrow \mathbf{0}^{|N(i)| \times 3}$
    for $k \in \{1..|N(i)|\}$ **do**
        $j \leftarrow N(i)_k$
        $(B_{km})_{m \in \{1..3\}} \leftarrow \Omega_{ijm}$
        $(B'_{km})_{m \in \{1..3\}} \leftarrow \Omega'_{ijm}$
    end for
    $K \leftarrow B'^{\mathrm{T}} B$ // Compute the covariance matrix.
    $\mathbf{U}, \mathbf{S}, \mathbf{V}^{\mathrm{T}} \leftarrow SVD(K)$ // Take the Singular Value Decomposition (SVD) of K.
    $\mathbf{s} \leftarrow sgn(det(\mathbf{U}\mathbf{V}^{\mathrm{T}}))$ // $sgn(\cdot)$ is the sign function.
    $\mathbf{S}' \leftarrow \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & \mathbf{s} \end{bmatrix}$ // To ensure $det(\mathbf{R}_i) > 0$.
    $(\mathbf{R}_{imn})_{m,n \in \{1..3\}} \leftarrow (\mathbf{U}\mathbf{S}'\mathbf{V}^{\mathrm{T}})_{mn}$
end for
// Step 3: Rotate each un-deformed neighborhood edge by the average of its connected vertices to get the rotated differential coordinates for each vertex.
$\mathbf{D} \leftarrow \mathbf{0}^{n_p \times 3}$
for $i \in \{1..n_p\}$ **do**
    for $k \in \{1..|N(i)|\}$ **do**
        $j \leftarrow N(i)_k$
        $\mathbf{J} \leftarrow \mathbf{R}_j + \mathbf{R}_i$ // $\mathbf{J} \in \mathbb{R}^{3 \times 3}$.
        $(\mathbf{D}_{im})_{m \in \{1..3\}} \leftarrow \mathbf{D}_{im} + \sum_{n \in \{1..3\}} [\mathbf{J}_{mn} \times \Omega_{ikn} \times (0.5 \times L_{ij})]$
    end for
end for
Return: $L\mathbf{D}$

The main idea of ARAP is to minimize the energy function (1) for each vertex i.

$$\mathcal{E}_i = \sum_{j \in N(i)} w_{ij} ||(p'_i - p'_j) - \mathbf{R}_i(p_i - p_j)||^2, \tag{1}$$

where w_{ij} is the Laplace weight of edge ij, p'_i is the new vertex position of i, p_i is the original vertex position of i. The ARAP alternates between updating all the p'_i and $\mathbf{R}_i$ ($\forall i \in n_p$) in each iteration. When p'_i are fixed, $\mathbf{R}_i$ are estimated to look for the best rigid rotation. When $\mathbf{R}_i$ are fixed, the function is minimized by solving the Laplace equation with edge ij rotated.

3 More Visualization Results

3.1 3D-to-2D Alignment

To demonstrate the effectiveness of our shape adjusting method (using differentiable ARAP), we show the union minus intersection map (as well as the IoU value) between the ground-truth 2D face segments and rendered face segments in Fig. 3.

3.2 Local Editing

The list of selected facial features used for editing is shown in Table 6. We adjust each controller separately to be -3σ and $+3\sigma$, then 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). More examples of our intuitive editing results are shown in Fig. 5.

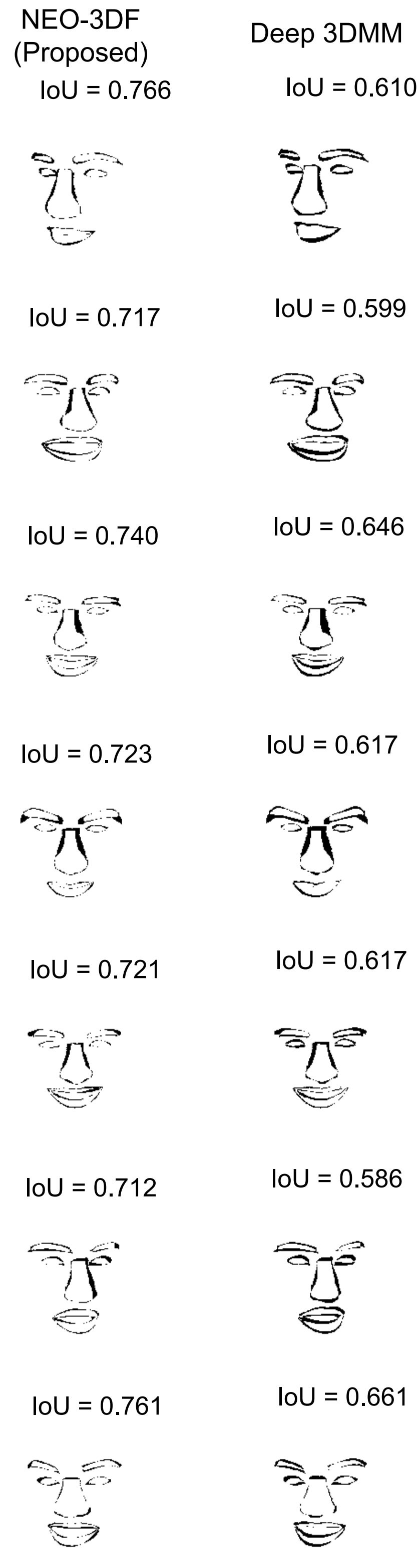

Fig. 3. 3D-2D segments alignment results.

Table 6. List of Facial Features Selected for Editing

Group	Feature Name	Editing Controller	Reference
Eyebrows	Curvature Strength	**EB-CS**	[6]
	Length	**EB-L**	
	Thickness	**EB-T**	
Eyes	Canthus Distance	**EY-CD**	[5, 6]
	Lateral Canthus (up/down)	**EY-LC**	
	Height	**EY-H**	
	Pupils Distance	**EY-PD**	
Nose	Bridge Width	**NS-BW**	[3, 4]
	Height	**NS-H**	
	Width	**NS-W**	
	Tip Depth	**NS-TD**	
	Tip Height	**NS-TH**	
	Tip Size	**NS-TS**	
Upper Lip	Height	**UL-H**	[4]
	Width	**UL-W**	
	Labial Fissure Width	**UL-LFW**	
Lower Lip	Height	**LL-H**	[4]
	Width	**LL-W**	
	End Height	**LL-EH**	
Rest	Facial Height	**RS-FH**	[4]
	Facial Width	**RS-FW**	
	Lower Facial Depth	**RS-LFD**	
	Lower Facial Height	**RS-LFH**	
	Mandibular Width	**RS-MW**	
	Middle Facial Depth	**RS-MFD**	
	Upper Facial Depth	**RS-UFD**	
	Upper Facial Height	**RS-UFH**	

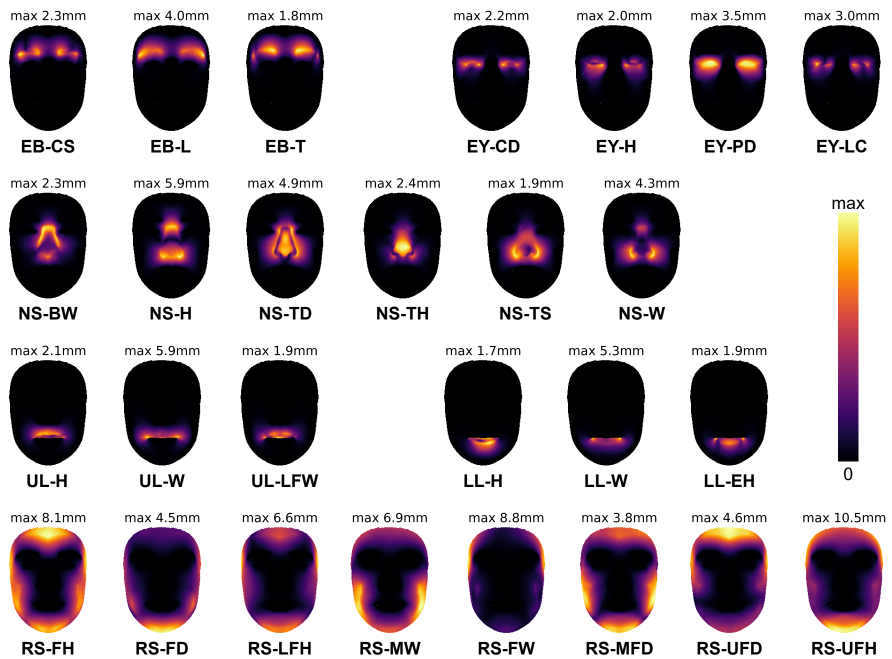

Fig. 4. Heatmaps of editing controllers.

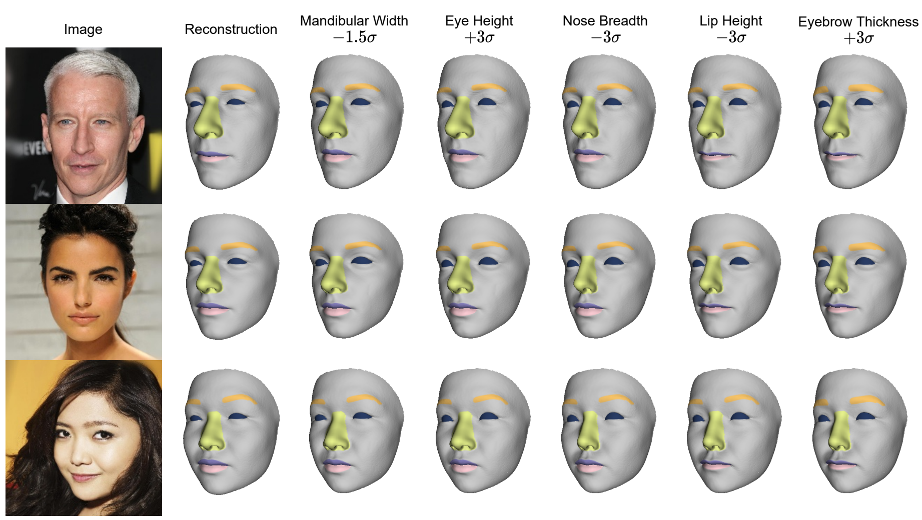

Fig. 5. Face editing demonstration.

References

1. Cao, Q., Shen, L., Xie, W., Parkhi, O.M., Zisserman, A.: Vggface2: A dataset for recognising faces across pose and age. In: 2018 13th IEEE international conference on automatic face & gesture recognition (FG 2018). pp. 67–74. IEEE (2018)
2. Esler, T.: Face recognition using pytorch (2020), https://github.com/timesler/facenet-pytorch
3. Farkas, L.G., Kolar, J.C., Munro, I.R.: Geography of the nose: a morphometric study. Aesthetic plastic surgery **10**(1), 191–223 (1986)
4. 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)
5. 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)
6. 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)
7. 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)
8. Sorkine, O., Alexa, M.: As-rigid-as-possible surface modeling. In: Symposium on Geometry processing. vol. 4, pp. 109–116 (2007)
9. Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., Rabinovich, A.: Going deeper with convolutions. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 1–9 (2015)