ASM: Adaptive Skinning Model for High-Quality 3D Face Modeling Supplementary Material

Kai Yang, Hong Shang[*], Tianyang Shi, Xinghan Chen, Jingkai Zhou, Zhongqian Sun, Wei Yang

Tencent AI Lab

{arvinkyang, hongshang, tirionshi, xinghanchen, fszhou, sallensun, willyang}@tencent.com

1. Model Implementation Details

We utilized Blender to place 84 bones on the face, as depicted in Fig. 1. The bone arrangement was derived from the distribution of the human skeletons and musculature to better represent the facial details of human faces. The orientations of all the bones are aligned parallel to the z-axis, except the root bone. The list of bone indices and names can be found in Tab. 1

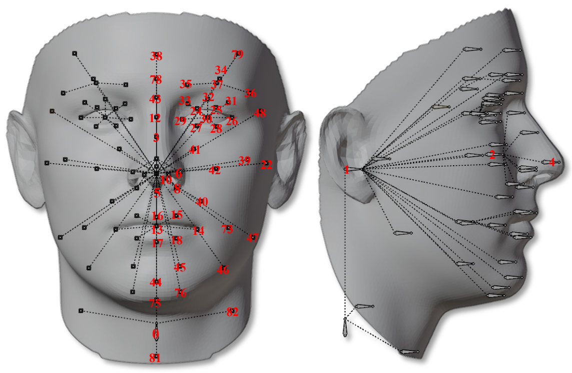

Figure 1. Illustration of the initial binding of ASM. The index numbers of bones on the left half of the face are displayed, while the index numbers of the right counterparts are omitted.

2. Dynamic Bone Binding

In the section of the main text on Dynamic Bone Binding, we discussed the process of updating the entire bone binding by modifying the UV coordinates of the bones. In the following section, we will provide additional elaboration on this topic.

Indices	Parents	Names	Indices	Parents	Names
0	-	root	42	head	apple_center.L
1	root	head	43	head	eyebrow_center
2	head	nose	44	head	chin
3	nose	nose_bridge	45	head	chin_side.L
4	nose	nose_tip	46	head	jaw.L
5	nose	nose_mid	47	head	jaw_corner.L
6	nose	nose_wing.L	48	head	temple.L
7	nose	nose_wing.R	49	head	ear.R
8	nose	nose_bottom.L	50	head	eye.R
9	nose	nose_bottom.R	51	eye.R	eye_inner_upper.R
10	nose	nose_hole.L	52	eye.R	eye_outer_upper.R
11	nose	nose_hole.R	53	eye.R	eye_outer_corner.R
12	nose	nose_bridge_upper	54	eye.R	eye_inner_lower.R
13	head	mouth	55	eye.R	eye_outer_lower.R
14	mouth	lip_corner.L	56	eye.R	eye_inner_corner.R
15	mouth	lip_upper_side.L	57	eye.R	eye_hole.R
16	mouth	lip_upper_mid	58	eye.R	eyelid_outer.R
17	mouth	lip_lower_mid	59	eye.R	eyelid_middle.R
18	mouth	lip_lower_side.L	60	eye.R	eyelid_inner.R
19	mouth	lip_corner.L	61	head	eyebrow.R
20	mouth	lip_upper_side.R	62	eyebrow.R	eyebrow_inner.R
21	mouth	lip_lower_side.R	63	eyebrow.R	eyebrow_outer.R
22	head	ear.L	64	eyebrow.R	eyebrow_mid.R
23	head	eye.L	65	head	apple_outer.R
24	eye.L	eye_inner_upper.L	66	head	apple_lower.R
25	eye.L	eye_outer_upper.L	67	head	apple_inner.R
26	eye.L	eye_outer_corner.L	68	head	apple_center.R
27	eye.L	eye_inner_lower.L	69	head	chin_side.R
28	eye.L	eye_outer_lower.L	70	head	jaw.R
29	eye.L	eye_inner_corner.L	71	head	jaw_corner.R
30	eye.L	eye_hole.L	72	head	temple.R
31	eye.L	eyelid_outer.L	73	head	cheek.L
32	eye.L	eyelid_middle.L	74	head	cheek.R
33	eye.L	eyelid_inner.L	75	head	chin_low
34	head	eyebrow.L	76	head	chin_side_low.L
35	eyebrow.L	eyebrow_inner.L	77	head	chin_side_low.R
36	eyebrow.L	eyebrow_outer.L	78	head	eyebrow_center_up
37	eyebrow.L	eyebrow_mid.L	79	head	forehead.L
38	head	forehead	80	head	forehead.R
39	head	apple_outer.L	81	root	neck_front
40	head	apple_lower.L	82	root	neck_side.L
41	head	apple_inner.L	83	root	neck_side.R

Table 1. Skeleton structure of ASM.

Barycentric Interpolation. We check whether the UV coordinate of bone j, denoted as ζ, falls within triangle f_{ABC}

[*]Corresponding author.

on UV space using the following formula:

$$\begin{aligned} t1 &= \overrightarrow{\zeta A} \times \overrightarrow{\zeta B} \\ t2 &= \overrightarrow{\zeta B} \times \overrightarrow{\zeta C} \\ t3 &= \overrightarrow{\zeta C} \times \overrightarrow{\zeta A} \end{aligned} \tag{1}$$

where $\times$ represents the cross-product operation. Point $\boldsymbol{\zeta}$ lies inside the triangle f_{ABC} when $t1$, $t2$, and $t3$ have the same sign ($t1 > 0$, $t2 > 0$, $t3 > 0$ or $t1 < 0$, $t2 < 0$, $t3 < 0$). We calculate the barycentric weights α, β, and γ of point $\boldsymbol{\zeta}$ within f_{ABC} with:

$$\alpha, \beta, \gamma = Barycentric(\boldsymbol{\zeta}, A, B, C) \tag{2}$$

where

$$\begin{aligned} \alpha &= \frac{-(x_\zeta - x_B)(y_C - y_B) + (y_\zeta - y_B)(x_C - x_B)}{-(x_A - x_B)(y_C - y_B) + (y_A - y_B)(x_C - x_B)} \\ \beta &= \frac{-(x_\zeta - x_C)(y_A - y_C) + (y_\zeta - y_C)(x_A - x_C)}{-(x_B - x_C)(y_A - y_C) + (y_B - y_C)(x_A - x_C)} \\ \gamma &= 1 - \alpha - \beta \end{aligned} \tag{3}$$

After obtaining the values of α, β, and γ, we use them to interpolate the coordinates of $\boldsymbol{\psi}$ in the world space. This is done based on the 3D coordinates of the three vertices, $\boldsymbol{v}_A$, $\boldsymbol{v}_B$, and $\boldsymbol{v}_C$. To recalculate $\boldsymbol{\psi}_j$ for bone j, we use the following formula:

$$\boldsymbol{\psi}_j = F'(\boldsymbol{\zeta}_j) = \alpha \mathbf{v}_A + \beta \mathbf{v}_B + \gamma \mathbf{v}_C - \mathbf{v}_t + \boldsymbol{\psi}_j^0 \tag{4}$$

where $\boldsymbol{\zeta}_j$ represents the UV coordinates, $\mathbf{v}_t$ represents the vertex closest to $\boldsymbol{\psi}_j^0$, and $\boldsymbol{\psi}_j^0$ represents the initial position of bone j.

Binding Updated. We provide a brief overview of the Linear Blend Skinning (LBS) method:

$$\mathbf{v}' = \sum_{j=1}^{J} w_j \mathbf{T}_j \mathbf{v} \tag{5}$$

The deformation is achieved through the use of w_j and $\mathbf{T}_j$, as demonstrated by Eq. 5. We expand $\mathbf{T}_j$ according to the following formula:

$$\begin{aligned} \mathbf{T}_j &= \mathbf{M}_j^{l2w} \mathbf{M}_j^{w2l} \\ &= \mathbf{M}_j^{l2w} \mathbf{B}_j^{-1} \\ &= \mathbf{M}_p^{l2w} \mathbf{M}^{trs}(\boldsymbol{\tau}_j) \mathbf{B}_j^{-1} \\ &= \prod_{p=1}^{P} \mathbf{M}^{trs}(\boldsymbol{\tau}_p) \mathbf{M}^{trs}(\boldsymbol{\tau}_j) \mathbf{B}_j^{-1} \end{aligned} \tag{6}$$

where $\mathbf{B}_j$ represents the bind-pose of bone j, and $\mathbf{M}^{trs}(\cdot)$ will be described in detail below. P denotes the parent chain for bone j. For example, the parent chain for nose_tip represents (nose_tip - nose - head - root). Eq. 6 shows that $\mathbf{T}_j$ has two parts: the bind-pose matrix for bone j that converts vertex $\mathbf{v}$ from the world space to the local space and the result of multiplying the transformation matrix of bone j with the local-to-world matrix of its parent bone, which converts $\mathbf{v}$ from the local space to the world space. When updating the 3D world position of bone j, it is necessary to update $\mathbf{B}_j$ and recalculate $\mathbf{M}^{trs}$ using the updated relative position between bones. The first step is to update the bind-pose matrix of each bone.

We keep rotation and scaling constant as in the initial binding, and only the translation component of the bind-pose matrix needs to be updated:

$$\mathbf{B}_j = B(\boldsymbol{\psi}_j) = \begin{bmatrix} R_j^0 S_j^0 & \boldsymbol{\psi}_j \\ 0 & 1 \end{bmatrix} \tag{7}$$

where R_j^0 and S_j^0 represent the rotation and scaling matrices identical to those present in the initial bind-pose matrix. $\boldsymbol{\psi}_j$ is the updated 3D world space position obtained from Eq. 4. Assuming that bones undergo just translation without rotation greatly simplifies the dynamic bone binding calculation stage.

We simply recalculate the new bind-pose matrix from the updated world coordinates of each bone. In the following, we will introduce how to calculate the transformation matrix by taking into account the updated relative positions between bones.

We define $\boldsymbol{\tau} = \left[\mathbf{r}^T, \mathbf{t}^T, \mathbf{s}^T\right]^T \in \mathbb{R}^9$ as the pose parameters in the local space of bone j, we decompose $\mathbf{M}^{trs}(\boldsymbol{\tau}_j)$ with:

$$\mathbf{M}^{trs}(\boldsymbol{\tau}_j) = \mathbf{M}_j^{l2p} \mathbf{M}(\boldsymbol{\tau}_j) \tag{8}$$

where $\mathbf{M}(\boldsymbol{\tau}_j) \in \mathbb{R}^{4\times4}$ is the standard transformation matrix composed from $\boldsymbol{\tau}_j$. $\mathbf{M}_j^{l2p}$ transform bone j into the coordinate system of its parent bone p, which can be solely determined by the bind-pose of the bones:

$$\mathbf{M}_j^{l2p} = \mathbf{B}_p^{-1} \mathbf{B}_j \tag{9}$$

where $\mathbf{B}_p$ and $\mathbf{B}_j$ are the bind-pose matrix of bone p and bone j, respectively.

With the introduction of these concepts, we complete the dynamic adjustment process for bone binding with Eq. 6. For details on the Dynamic Bone Binding process, refer to Algorithm 1.

3. Face Registration

We conducted face registration experiments on LYHM [3] and FaceScape [14] to evaluate the representation capacity of different parametric face models. We marked 7 key points on the ground-truth scans, consistent with the NoW-Benchmark prototype [13]. We identified an

Algorithm 1: Dynamic Bone Binding

```
Input: $\boldsymbol{\zeta}, \boldsymbol{\tau}, \boldsymbol{\psi}^0, \mathbf{v_t}, \mathbf{B}^0$
    // barycentric interpolation phase
 1: for bone $j$ in skeleton do
 2:     for triangle $f$ in UV map do
 3:         if $\boldsymbol{\zeta}_j$ falls within $f$ based on Eq. 1 then
 4:             Update $\alpha, \beta, \gamma$ with $f$ based on Eq. 2
 5:             Update $\boldsymbol{\psi}_j$ with $\alpha, \beta, \gamma$ based on Eq. 4
 6:         end if
 7:     end for
 8: end for

    // binding updated phase
 9: for bone $j$ in skeleton do
10:     Update $\mathbf{B}_j$ with $\boldsymbol{\psi}_j$ and $\mathbf{B}_j^0$ based on Eq. 7
11:     Update $\mathbf{M}_j^{l2p}$ with $\mathbf{B}_j$ and $\mathbf{B}_p$ based on Eq. 9
12:     Update $\mathbf{M}^{trs}(\boldsymbol{\tau}_j)$ with $\boldsymbol{\tau}_j$ and $\mathbf{M}_j^{l2p}$ based on Eq. 8
13: end for
14: for bone $j$ in skeleton do
15:     Update $\mathbf{T}_j$ with $\mathbf{M}^{trs}(\boldsymbol{\tau}_j)$, $\prod_{p=1}^{P} \mathbf{M}^{trs}(\boldsymbol{\tau}_p)$ and $\mathbf{B}_j$
        based on Eq. 6
16: end for
Output: $\mathbf{T}$
```

equivalent number of key points on the model's topology and utilized the 3D coordinates of 7 key point pairs to compute global rigid transformation parameters as initial values for fitting. We used the $point\_mesh\_face\_distance()$ function from PyTorch3D [12] to optimize the distance between the ground-truth scan vertices and the nearest triangle on the predicted mesh. We compared our model with BFM [10], FLAME [8], CoMA [11] , FaceScape [14], ImFace [15], and MetaHuman [5].

BFM uses a 199-dimensional shape basis and a 79-dimensional expression basis in its optimization process. FLAME employs a 300-dimensional shape basis, a 100-dimensional expression basis, and rotation parameters for two joints, neck and chin, total of 406 parameters. We retrained the encoder and decoder networks of CoMA with 64-dimensional latent vectors on the CoMA datasets. During fitting, we only optimized the latent vector while fixing the decoder network parameters. FaceScape uses a 300-dimensional shape basis and a 51-dimensional expression basis, excluding the natural expression from the original 52-dimensional expression basis. To preserve the mesh's initial scale, we subtracted the sum of the remaining 51 dimensions to obtain the value of the natural expression basis. This approach ensured that the sum of all 52 expression dimensions equaled 1, thereby reducing one degree of freedom. For ImFace, we converted the ground truth mesh to SDF and fitted the corresponding z_id and z_exp using a pre-trained network. We then converted the SDF results to meshes and evaluated them in our metric-calculation prototype. For MetaHuman, we fitted all 9 degrees of freedom of the bones to deform the meshes.

We utilized Adam optimizer in PyTorch [9] to optimize the model parameters and the global rigid transformation parameters, minimizing the point-to-face distance, while adding regularization to avoid model artifacts. We kept the same learning rate of 1e-3 and iteration steps of 300 for all models, except for ImFace, for which we used the default iteration steps of 1,500 as used in the released code.

4. Multi-view Face Reconstruction

Compared to Multi-view Stereo (MVS) without using the face priors, parametric face models greatly reduce the number of parameters to be solved, which alleviated the underdetermined problem. We solved the multi-view face reconstruction with a parametric face model as an optimization problem with constraints from multiple views. For all the parametric face models in the experiments, we used the predicted results from a single-view face reconstruction network as the initial point for quick and stable convergence of the optimization process.

4.1. Single-view Prediction Network

We followed Deng *et al.* [4] and built a reconstruction network based on 3DMM. Given an image I captured from one of the multiple views, we used a neural network to predict its parameter set $X = (\boldsymbol{\alpha}, \boldsymbol{\beta}, \mathbf{t}, \mathbf{R})$, where $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ represented the shape and expression parameters of the 3DMM basis, respectively. The 3D face shape $\boldsymbol{S}$ could be represented by $\mathbf{S} = \bar{\mathbf{S}} + \mathbf{B}_{id}\boldsymbol{\alpha} + \mathbf{B}_{exp}\boldsymbol{\beta}$. The translation $\mathbf{t} \in \mathbb{R}^3$ and rotation $\mathbf{R} \in SO(3)$ represented the pose of the 3D model with respect to the current camera position. We used a perspective projection model to obtain the projected coordinates $\mathbf{p}$ of the vertices $\mathbf{v}$ in image I as $\mathbf{p} = \Pi(\mathbf{Rv} + \mathbf{t})$, where $\Pi(\cdot)$ represented the perspective projection.

To obtain an initial shape for multi-view optimization, we simply averaged the predicted $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ across multiple views to obtain the initial face mesh.

$$\mathbf{S} = \bar{\mathbf{S}} + \mathbf{B}_{id}\boldsymbol{\alpha}_{mean} + \mathbf{B}_{exp}\boldsymbol{\beta}_{mean} \tag{10}$$

To use the predicted face shape, an additional registration step is required to get initial model parameters to fit the predicted mesh. For BFM and ASM with the same topology as Deng *et al.* [4], the point-to-point loss was employed. For FLAME and MetaHuman with different topologies, point-to-face loss as described in Sec. 3 was used. For all the models, the registration step was implemented with Adam optimizer with a learning rate of 1e-3 and iteration steps of 300. These model parameters, together with pose parameters, would be solved in the following optimization step for multi-view reconstruction.

4.2. Energy Function

Landmarks. We used the facial key points prediction method [2] to predict the positions of 68 facial key points from the input image. We annotated the corresponding 68 vertices on the mesh and obtained the projected coordinates of these vertices on the corresponding image with the perspective projection function. We then minimized the landmarks term:

$$E_{lmk}(\hat{\mathbf{v}}) = \frac{1}{N}\sum_{n=1}^{N} ||\mathbf{q}^n - \Pi(\mathbf{R^n}\hat{\mathbf{v}} + \mathbf{t}^n)||^2 \tag{11}$$

where $\mathbf{q}^n$ were the key points from landmarks prediction, and $\hat{\mathbf{v}}$ were the masked 68 vertices.

Regularization. We imposed constraints on the parameters to be optimized in the parametric face models to prevent over-fitting. For the ASM model parameters $\boldsymbol{\tau}, \boldsymbol{\zeta}, \boldsymbol{\pi}, \boldsymbol{\mu}, \boldsymbol{\Sigma}$, we used the following shape priors:

$$\begin{aligned} E_{reg}(\boldsymbol{\tau}, \boldsymbol{\zeta}, \boldsymbol{\pi}, \boldsymbol{\mu}, \boldsymbol{\Sigma}) =& \lambda_1||\boldsymbol{\tau}||^2 + \lambda_2||\boldsymbol{\zeta}||^2 + \lambda_3||\boldsymbol{\pi} - \boldsymbol{\pi}'||^2 + \\ & \lambda_4||\boldsymbol{\mu} - \boldsymbol{\mu}'||^2 + \lambda_5||\boldsymbol{\Sigma} - \boldsymbol{\Sigma}'||^2 \end{aligned} \tag{12}$$

where $\boldsymbol{\pi}', \boldsymbol{\mu}', \boldsymbol{\Sigma}'$ were the initial parameters of ASM from fitting the skinning weights generated by Blender. Here we use $\lambda_1 = 1, \lambda_2 = \lambda_3 = \lambda_4 = \lambda_5 = 0.1$

Edge. Due to the recent development of deep learning-based face parsing algorithms, we could easily obtain a refined segmentation of facial semantic area. We trained a face segmentation network on the CelebaMask-HQ [7] datasets, and with an image I we predicted its segmentation mask and extracted the face contour points set donated as E for constraining the mesh's contour.

To create the contour points set of the reconstructed face model, we applied a rigid transformation with the predicted pose parameters. We calculated the normal direction of each vertex on the model surface and computed the absolute value of the normal on the z-axis. We considered the z-axis absolute value of the vertices normal less than a threshold to be the points at the edge of the face model and included them in the candidate set.

To ensure that only visible vertices were used, we utilized z-buffering to determine whether the vertices in the candidate point set are visible. This enabled us to screen out the final contour vertices. Finally, we minimized the edge term by extracting contour points from both the image and the model contour points set.

$$\begin{aligned} E_{edge}(\mathbf{v}') &= \frac{1}{N}\sum_{n=1}^{N} chamfer(E^n, \Pi(\hat{\mathbf{v}}^n)) \\ \hat{\mathbf{v}} &= \{\mathbf{v}' | |N(\mathbf{v}')_z| < \theta \; and \; V(\mathbf{v}') > 0\} \end{aligned} \tag{13}$$

where $\mathbf{v}' = \mathbf{R^n}\mathbf{v} + \mathbf{t}^n$ and θ was the threshold used for distinguishing edge vertices. $N(\cdot)$ was the normal calculation function for vertices and $V(\cdot)$ was the z-buffering test to decide whether vertex $\mathbf{v}$ was visible in the current frame, respectively. In experiments, we used θ of $10°$.

Photometric Consistency. The photometric consistency constraint method has found wide application in solving the parameters of face models from multi-view images [1, 6]. The technique involves projecting the vertices of the mesh onto images from various angles and sampling the corresponding pixel intensities on the original image using the projection coordinates $\mathbf{p}$. Hernandez *et al.* [6] utilizes a 3x3 intensity patch centered around $\mathbf{p}$ to use the photometric consistency. However, we have observed that this approach is not ideal when the original vertex $\mathbf{v}$ lies on a plane with significant depth variation. In such cases, the 3x3 patch includes pixel information with a broad depth range, resulting in less accurate photometric consistency constraints.

To address this limitation, we projected and sampled the intensity values for each vertex $\mathbf{v}_j$ on the model and obtain $S_j = \Gamma(\Pi(\mathbf{R}\mathbf{v}_j + \mathbf{t}))$, where $\Gamma(\cdot)$ was the interpolation function for sample the intensity on the projection coordinate of $\mathbf{v}_j$. We then unwrapped the intensities of each vertex to the UV space to obtain the UV-intensities map $\mathbf{U} = Unwrap(\mathbf{S})$.

In the UV space, we performed Local Normalized Cross-correlation (LNCC) on the UV map,

$$\begin{aligned} & LNCC(\mathbf{U^i}, \mathbf{U^j}) \\ & = \frac{1}{X}\sum_{x=1}^{X} V(S_x^i)V(S_x^j)NCC(P(U_x^i), P(U_x^j)) \end{aligned} \tag{14}$$

where $V(S_x^i)V(S_x^j)$ were the same z-buffering test function to decide the visible mask of points S_x^i and S_x^j, respectively. X were the total number of vertices. $P(U_x^i)$ is the 3x3 sampling function around pixel U_x^i, and NCC was the Normalized Cross-correlation function. The LNCC function captured the photometric information on the UV space rather than the image space. This process generated a sliding window that conformed to the mesh surface of the model, accurately sampled the vertex intensities even in areas with significant depth variation, and reduced the depth of field effect caused by sampling at a large scale.

We calculated the LNCC loss on the unwrapped UV maps between any two frames in the input multi-view images. We then minimized the photometric-consistency term:

$$E_{pc}(\mathbf{v}) = \frac{2}{N(N-1)}\sum_{i=1}^{N}\sum_{j=i+1}^{N} LNCC(\mathbf{U}^i, \mathbf{U}^j) \tag{15}$$

4.3. Multi-view Optimization

We finally sum up all the energy terms and optimized the parameters of ASM as:

$$E_{total} = \lambda_1 * E_{lmk} + \lambda_2 * E_{edge} + \lambda_3 * E_{pc} + \lambda_4 * E_{reg} \quad (16)$$

where $\lambda_1 = 0.001$, $\lambda_2 = 0.4$, $\lambda_3 = 100$ and $\lambda_4 = 1$ in our experiments. We utilized Adam optimizer to optimize the parameters of the model and the pose, minimizing the energy term of Eq. 16. The same learning rate of 1e-4 and iteration steps of 500 were used for all the models.

References

[1] Brian Amberg, Andrew Blake, Andrew Fitzgibbon, Sami Romdhani, and Thomas Vetter. Reconstructing high quality face-surfaces using model based stereo. In *2007 IEEE 11th International Conference on Computer Vision*, pages 1–8. IEEE, 2007.

[2] Adrian Bulat and Georgios Tzimiropoulos. How far are we from solving the 2d & 3d face alignment problem?(and a dataset of 230,000 3d facial landmarks). In *Proceedings of the IEEE international conference on computer vision*, pages 1021–1030, 2017.

[3] Hang Dai, Nick Pears, William Smith, and Christian Duncan. Statistical modeling of craniofacial shape and texture. *International Journal of Computer Vision*, 128:547–571, 2020.

[4] Yu Deng, Jiaolong Yang, Sicheng Xu, Dong Chen, Yunde Jia, and Xin Tong. 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*, pages 0–0, 2019.

[5] Epic Games. Metahuman creator. *Available: https://www.unrealengine.com/en-US/metahuman-creator*, 2021.

[6] Matthias Hernandez, Tal Hassner, Jongmoo Choi, and Gerard Medioni. Accurate 3d face reconstruction via prior constrained structure from motion. *Computers & Graphics*, 66:14–22, 2017.

[7] Cheng-Han Lee, Ziwei Liu, Lingyun Wu, and Ping Luo. Maskgan: Towards diverse and interactive facial image manipulation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[8] 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.

[9] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32, 2019.

[10] Pascal Paysan, Reinhard Knothe, Brian Amberg, Sami Romdhani, and Thomas Vetter. A 3d face model for pose and illumination invariant face recognition. In *2009 sixth IEEE international conference on advanced video and signal based surveillance*, pages 296–301. Ieee, 2009.

[11] Anurag Ranjan, Timo Bolkart, Soubhik Sanyal, and Michael J Black. Generating 3d faces using convolutional mesh autoencoders. In *Proceedings of the European conference on computer vision (ECCV)*, pages 704–720, 2018.

[12] Nikhila Ravi, Jeremy Reizenstein, David Novotny, Taylor Gordon, Wan-Yen Lo, Justin Johnson, and Georgia Gkioxari. Accelerating 3d deep learning with pytorch3d. *arXiv preprint arXiv:2007.08501*, 2020.

[13] Soubhik Sanyal, Timo Bolkart, Haiwen Feng, and Michael J Black. Learning to regress 3d face shape and expression from an image without 3d supervision. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7763–7772, 2019.

[14] 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.

[15] Mingwu Zheng, Hongyu Yang, Di Huang, and Liming Chen. Imface: A nonlinear 3d morphable face model with implicit neural representations. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 20343–20352, 2022.

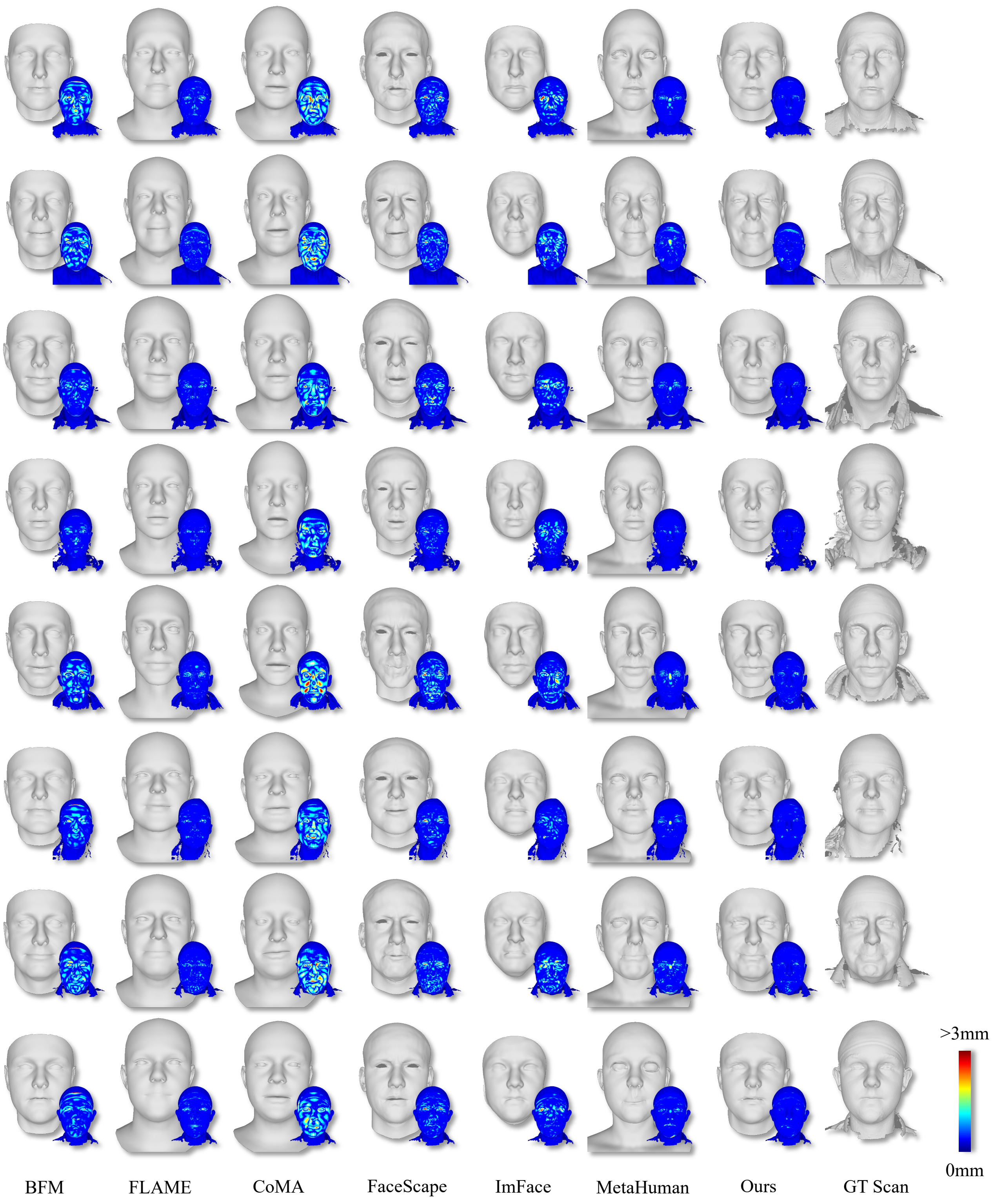

Figure 2. More examples of fitting result on LYHM [3]. GT Scans stand for the ground truth scan used for fitting.

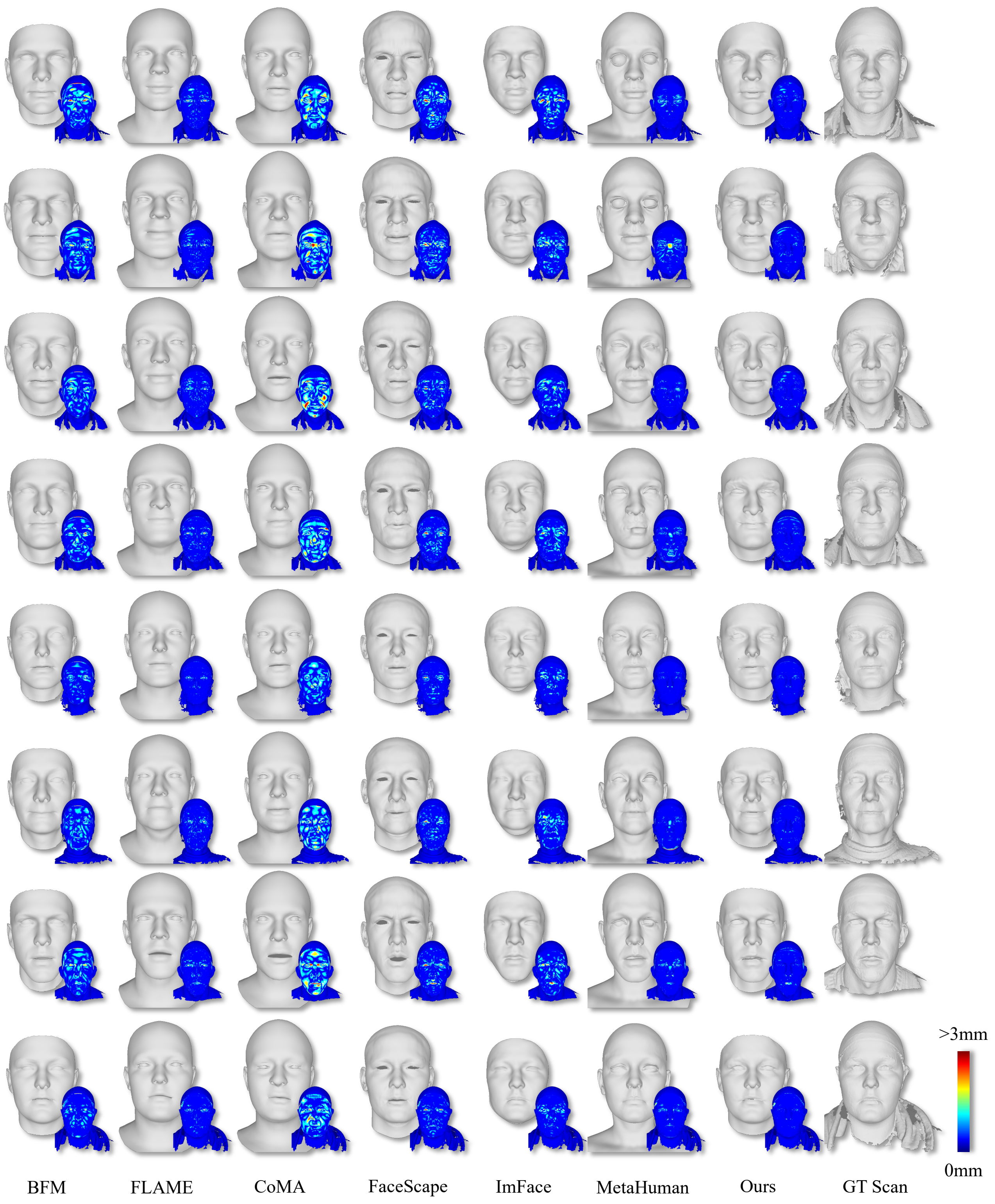

Figure 3. More examples of fitting result on LYHM [3]. GT Scans stand for the ground truth scan used for fitting.

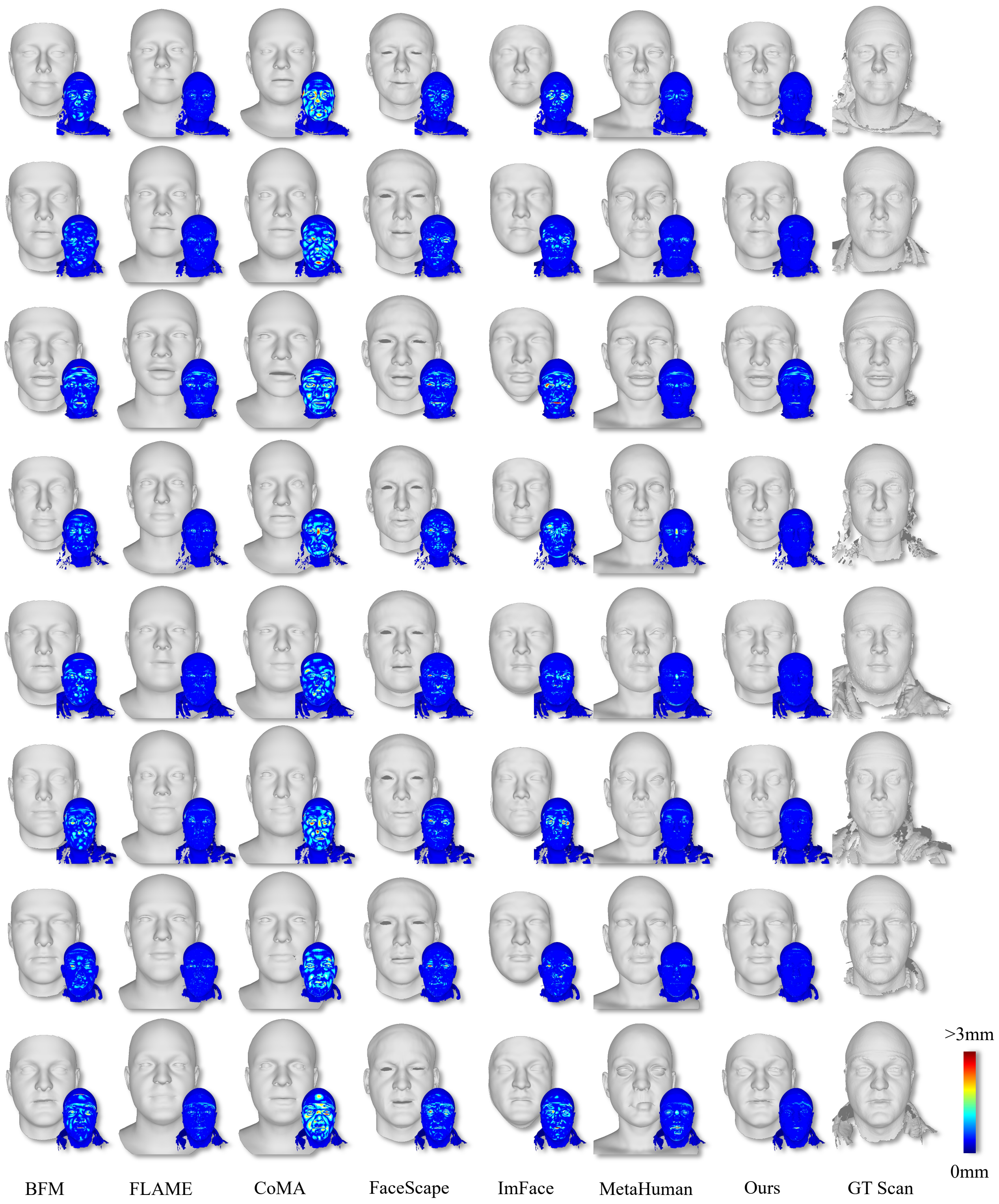

Figure 4. More examples of fitting result on LYHM [3]. GT Scans stand for the ground truth scan used for fitting.

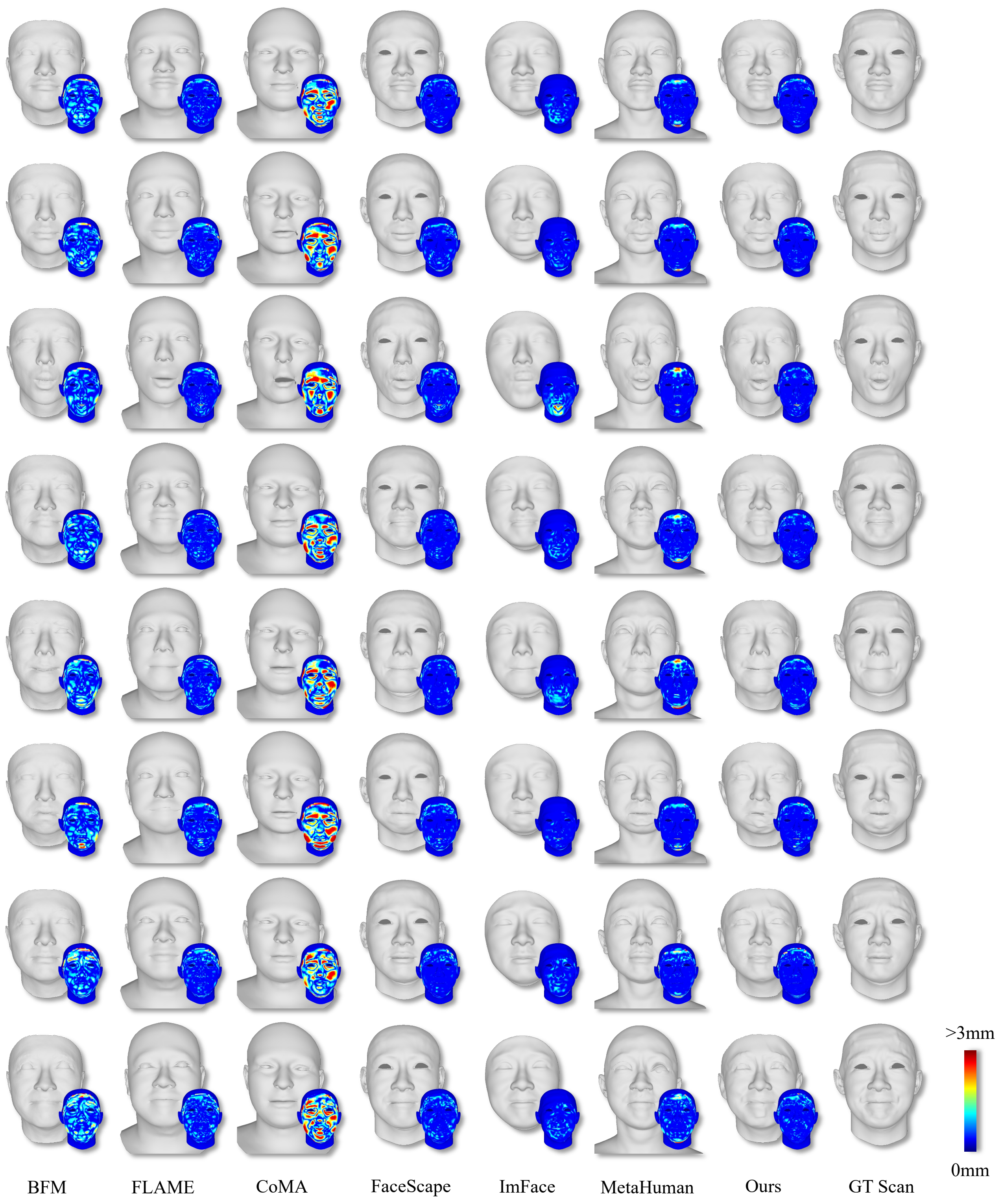

Figure 5. More examples of fitting result on FaceScape [14] . GT Scans stand for the ground truth scan used for fitting.