Supplemental - Representing 3D Faces with Learnable B-Spline Volumes

Prashanth Chandran Daoye Wang Timo Bolkart
Google
{prchandran, daoye, tbolkart}@google.com

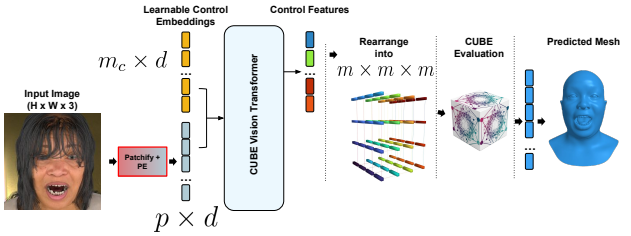

Figure 1. CUBE control features can be regressed from images by concatenating m_c learnable control embeddings to image patch tokens, and processing them with a standard vision transformer. With the exception of the *patchify* layer before the transformer encoder, our architecture for image-based regression is identical to the model we introduced for scan registration in the main paper. For image-based regression, we used a ViT-Large backbone with $8 \times 8 \times 8$ control tokens.

1. Additional Implementation Details

1.1. Image-based Regression

Architecture. As we briefly mentioned in the main paper, CUBE can also be used as the output representation when regressing 3D faces from an RGB image. To achieve this, we pass an input image $(H \times W \times 3)$ through a standard *patchify* layer and obtain patch tokens of shape $(p \times d)$. A learnable position encoding is applied to these patch tokens [1]. Similar to the CUBE scan encoder, we append m_c learnable control token embeddings; each of size d, along the sequence length of the p position encoded patch tokens. We then use a standard vision transformer [1] to regress CUBE control features from the input collection of $(p+m_c)$ tokens as shown in Fig. 1. The vision transformer outputs the control features that are evaluated using CUBE as shown in figure 2 of the main paper.

Dataset. We use a synthetic dataset of 500K portraits rendered at a resolution of 224×224 to train our image-to-CUBE model. This synthetic dataset was generated using the same process described in section 4.1 of the main paper. As we are regressing the shape directly from a single input image, we only render an accessorized 3D head from a single randomly placed camera and skip the scan reconstruction step.

Training. We use a ViT-Large model as our encoder and initialize its weights from the official MAE's encoder checkpoint [2] to speed up convergence. We use $8 \times 8 \times 8$ control features for CUBE. We apply standard photometric and geometric augmentations on the input images during training. We train our model with an L1 loss on the predicted shape. We use a batch size of 128 and train the model for 100,000 steps on 8 TPUs using the AdamW optimizer [3] and a learning rate of 1e-4.

2. Additional Results

2.1. Performance-cost trade off

Table 1 of the main paper confirms a tradeoff between control point density and the level of detail added by the residual mlp g. With fewer control points (4^3), the difference in the point-to-scan distance between predictions w/ and w/o the refinement MLP is 3-7x larger, depending on the encoder size. The inference times (measured on a V100 GPU) of our models with a varying number of control points ranges from 0.20s (CUBE-S, 4^3) to 2.92s (CUBE-L, 16^3).

2.2. Sampling robustness for scans

Our model for scan registration is robust to varying sample counts and sampling strategies. In Fig. 2, we sample the same input scan three times to obtain point clouds of different sizes consisting of 25,000, 50,000 and 75,000 points respectively. The sampled point clouds are processed with CUBE-L model with $16 \times 16 \times 16$ control points to obtain the corresponding registered meshes. As we see in Fig. 2, the CUBE scan registration model produces perceptually similar outputs with only minor differences in face shape.

2.3. Effect of position encoding

In Fig. 3, we compare the training curves of two identical scan-to-mesh models w/ and w/o position encoding (PE) [4] to show its effect on the convergence. We used a CUBE-Large model with $16 \times 16 \times 16$ controls for this experiment.

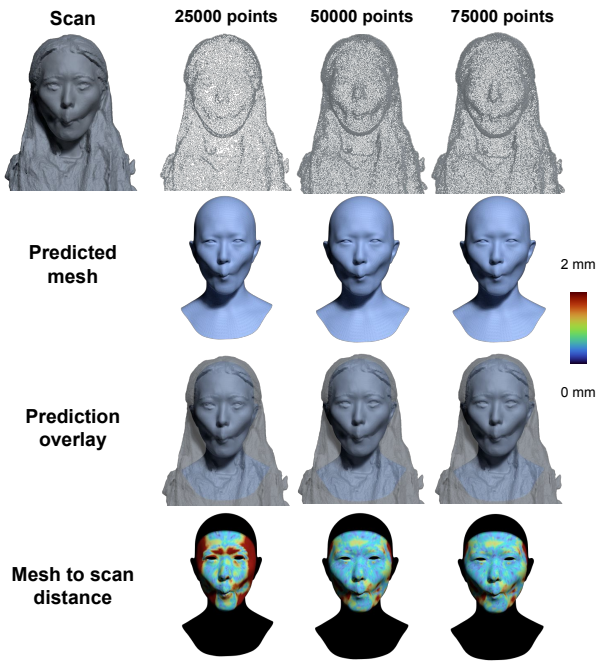

Figure 2. Given a high resolution scan, we randomly sample 25K, 50K and 75K points from the scan to result in point clouds of different sizes as seen in the first row. These sampled point clouds are independently processed by our CUBE-L model to result in the predictions seen in the second row. The third row overlays the predicted meshes on the original scan. The last row shows the mesh to scan distance for the predicted meshes. The predicted shapes remain plausible and capture the geometry of the scan in all cases demonstrating that our model is able to gracefully handle point clouds of different sizes.

Figure 3. Position encoding (PE) the input scan points has a positive effect on the convergence of our CUBE encoder. While the difference in final training error is not significant, PE did seem to accelerate convergence in the earlier iterations of training.

While we did not observe a significant difference in the final training error, we noticed that PE speeds up the convergence of our model in the earlier stages of training.

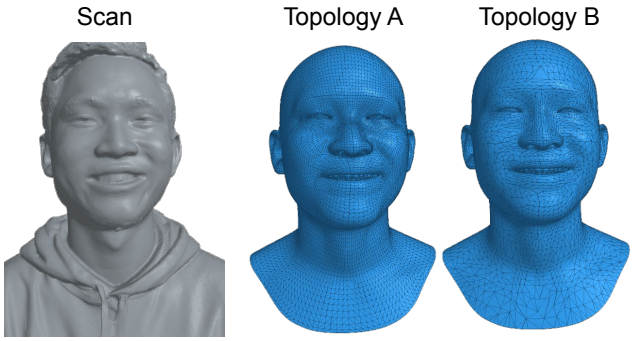

Figure 4. We show an example where by sampling CUBE at different locations on the template mesh surface, we can dictate the topology of the output shape. When combined with our transformer based encoder for scan registration, this allows CUBE to be used not only with input scans with a varying number of points, but also to represent output meshes in different topologies.

2.4. Topology changes

CUBE is a hybrid representation of geometry that can be evaluated continuously like an implicit representation, while still maintaining correspondence with a template mesh. Therefore CUBE can produce meshes in arbitrary output topologies without any retraining. As we saw in figure 2 of the main paper, CUBE is evaluated in two stages. First, we evaluate a B-Spline volume using the predicted high dimensional control features at locations sampled from a normalized template mesh. This is followed by evaluating a point-wise residual MLP to obtain residual geometric details. By changing the topology of the template shape at inference, we can control the topology in which the output mesh is generated without having to re-train CUBE. In Fig. 4, we show an example of a predicting a registered mesh in two different output topologies from an input scan using the same model. This flexibility makes CUBE a very useful geometric representation in practice.

2.5. Optimizing CUBE representations

CUBE as a representation can also be optimized or fit to target constraints without an encoder. We present two examples for optimization-based fitting with CUBE.

Fitting to a mesh. Given a target mesh with proxy surfaces for the scalp and the beard (see Fig. 5 left), our objective is to optimize for CUBE's latent control features $\mathbf{c}_{ijk}$ along with the residual MLP g to approximate this target mesh as closely as possible. We use $4 \times 4 \times 4$ control features for this experiment where the dimension of each control feature is 16. We randomly initialize the CUBE control features and the MLP g and optimize them to minimize the vertex to vertex distance to the target scan using gradient descent. We use the adamw [3] optimizer with a learning rate of 1e-3 and optimize for 2000 steps. As seen in Fig. 5 (left), CUBE can

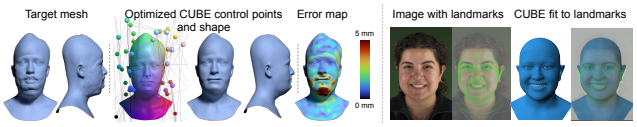

Figure 5. Left: We show how CUBE can be optimized to match the shape of a target mesh with proxy surfaces for the scalp and facial hair. The optimized control points along with the final mesh and the approximation error are also shown. Right: We show the result of fitting CUBE to match detected dense landmarks on an input image. CUBE can reasonably approximate the facial geometry with a small number of $4 \times 4 \times 4$ control features in both cases.

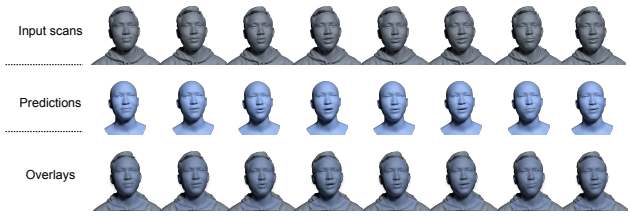

Figure 6. The CUBE scan encoder produces temporally smooth results for input scan sequences and can be used for performance registration. In this figure, we show the input scan, the prediction of our model and an overlay of the prediction on the input scan for a sequence of scans reconstructed from a facial performance of a subject. Note that the scans were processed independently for each frame.

approximate the facial geometry with proxy surfaces with only a small number of control points.

Fitting to 2D landmarks. We can optimize CUBE to fit detected 2D landmarks on images using a standard landmark re-projection energy (see Fig. 5 right). Given a portrait image of a subject, we run a dense landmark detector to detect 600 landmarks in the image. We optimize for the CUBE control features in camera space so that the recovered shape when projected on the image, matches the detected 2D landmarks. We assume that the intrinsics are fixed for this experiment. The optimization parameters are the same as for the mesh fitting experiment, and we solve for $4 \times 4 \times 4$ control features of dimension 16 using gradient descent. We optimize with a learning rate of 1e-2 for 500 steps.

2.6. Temporal predictions

Our model produces temporally smooth predictions when applied to scan and image sequences. In Fig. 6 and Fig. 7 we show predictions from our model for sequential scan and image data respectively. Kindly have a look at our supplemental video for more results.

2.7. Qualitative results

Finally we provide additional qualitative results for our scan registration and image-based face reconstruction models

Figure 7. The CUBE image encoder can process frames from a video sequence to reconstruct facial performances in the wild. In this example, we show the captured images in the first row and the corresponding predictions from our model for each image in the second row. Our model produces a faithful reconstruction of the input facial performance.

in Fig. 8 and Fig. 9 respectively. These results highlight CUBE's ability to represent diverse face shapes and expressions while also being a continuous and localized shape representation.

References

[1] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. *ICLR*, 2021. 1

[2] Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross Girshick. Masked autoencoders are scalable vision learners. *arXiv:2111.06377*, 2021. 1

[3] Ilya Loshchilov and Frank Hutter. Fixing weight decay regularization in adam. *CoRR*, abs/1711.05101, 2017. 1, 2

[4] Matthew Tancik, Pratul P. Srinivasan, Ben Mildenhall, Sara Fridovich-Keil, Nithin Raghavan, Utkarsh Singhal, Ravi Ramamoorthi, Jonathan T. Barron, and Ren Ng. Fourier features let networks learn high frequency functions in low dimensional domains. In *Advances in Neural Information Processing Systems (NeurIPS)*, Red Hook, NY, USA, 2020. Curran Associates Inc. 1

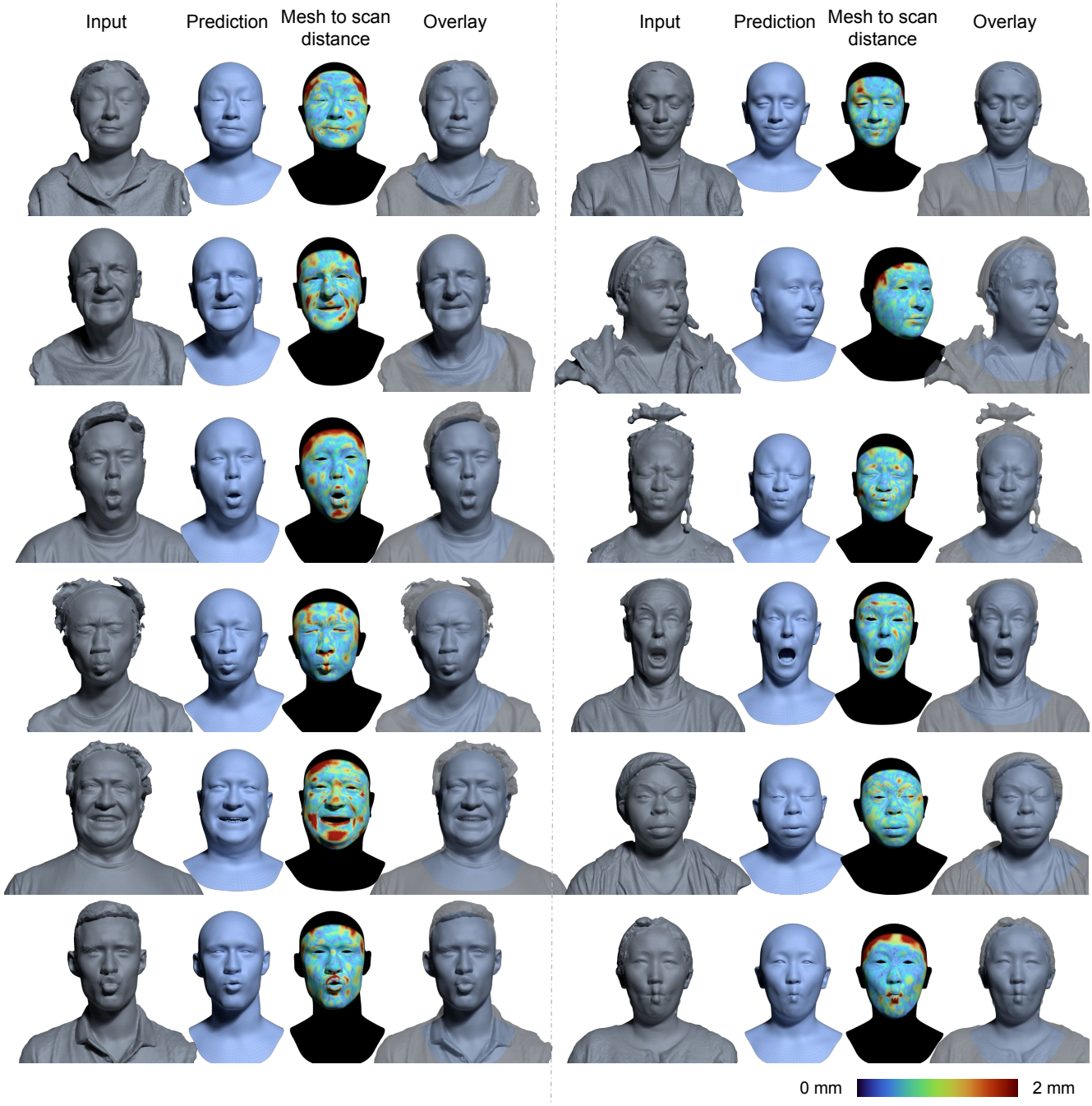

Figure 8. Additional qualitative results from our CUBE-Large $16 \times 16 \times 16$ model for scan registration. For each input scan, we visualize the prediction from the model and the point to scan distance (mm). Our method can handle input scans in varying topologies acquired from a diverse collection of subjects in various expressions and accessories.

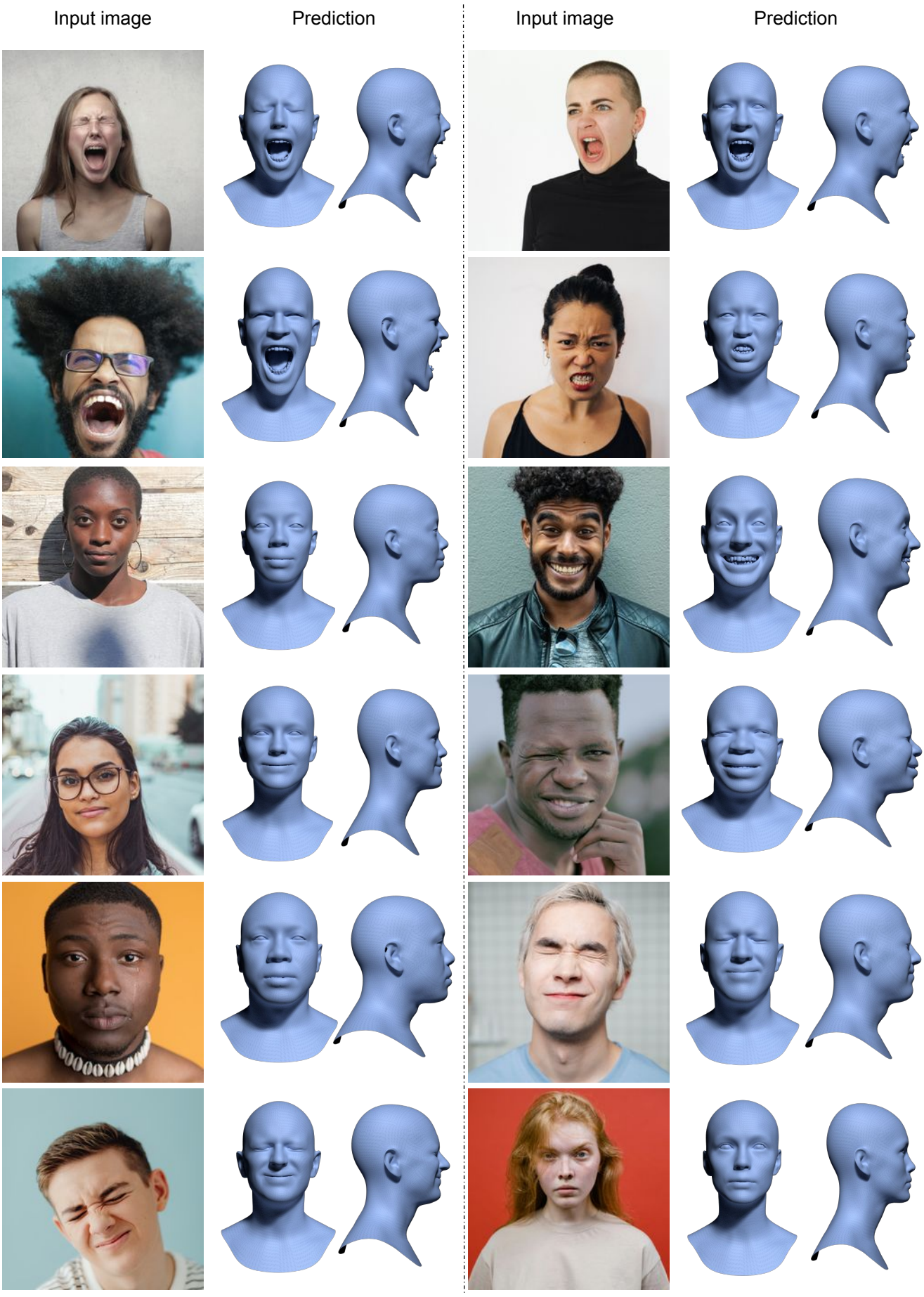

Figure 9. Additional qualitative results from our CUBE-Large $8 \times 8 \times 8$ model for face reconstruction from monocular images. For each input image, we visualize the predicted from the front and the side.