EUROGRAPHICS 2024 / A. Bermano and E. Kalogerakis
(Guest Editors)

COMPUTER GRAPHICS *forum*
*Volume 43 (2024), Number 2*

# Learning to Stabilize Faces

## Appendix

J. Bednarik and E. Wood and V. Choutas and T. Bolkart and D. Wang and C. Wu and T. Beeler

Google

We expand on the details behind sampling random rigid transformations in Sec. 1, and we provide details of training the modified prior work of [WSS18] in Sec. 2.

### 1. Sampling Random Rigid Transformations

The Algorithm 1 in the main paper describes the process of generating the misaligned pairs of face meshes to train our method. We introduced the function $\mathcal{G}(\varepsilon_R, \varepsilon_T) : \mathbb{R} \times \mathbb{R} \rightarrow \mathbb{R}^{4\times4}$ which, given two scalars $\varepsilon_R, \varepsilon_T$, generates a random rigid transformation

$$S_\varepsilon = \begin{bmatrix} R & \mathbf{t} \\ \mathbf{0}^\top & 1 \end{bmatrix},$$

where $R \in \mathbb{R}^{3\times3}$ is a rotation matrix and $\mathbf{t} \in \mathbb{R}^3$ is a translation vector.

We generate the rotation matrix $R$ by sampling an angle $\alpha$ from a normal distribution parameterized by $\varepsilon_R$ and a random axis $\mathbf{a} \in \mathbb{R}^3$ as follows:

$$\begin{aligned} \alpha &\sim \mathcal{N}(0, \varepsilon_R) \\ x, y, z &\sim \mathcal{U}(-1, 1) \\ \mathbf{a} &= [x, y, z]^\top \\ S_\varepsilon &= \mathcal{A}(\alpha, \mathbf{a}), \end{aligned}$$

where $\mathcal{A}$ converts an angle and an axis to the rotation matrix. Finally, we generate the translation vector $\mathbf{t}$ as follows:

$$\begin{aligned} x, y, z &\sim \mathcal{N}(\mathbf{0}, \varepsilon_T) \\ \mathbf{t} &= [x, y, z]^\top. \end{aligned}$$

### 2. Learned Confidence Map Implementation

As discussed in Section 4.3. of the main paper, we re-implemented the existing work of [WSS18], referred to as CMAP, but found the original formulation to produces unsatisfactory results. This section details the modification and training procedure we applied to boost its performance.

At its core, CMAP performs a Procrustes alignment. However, the strength of the methods come from the learned facial mask, which determines the facial regions to be considered for the rigid alignment. Let $U_s, U_t \in \mathbb{R}^{4\times N}$ be the source and target vertices in homogeneous coordinates, $\mathbf{w} \in [0, 1]^N$ per-vertex weights, $\mathcal{P}(U_s, U_t)$ Procrustes alignment producing the source vertices aligned to the target, and $\odot$ Hadamard product. We can compute weighted source and target vertices as

$$\begin{aligned} \tilde{U}_s &= W \odot U_s \\ \tilde{U}_t &= W \odot U_t \\ W &= [1, 1, 1, 1]^\top \mathbf{w}^\top. \end{aligned}$$

Then, CMAP finds the optimal weights $\mathbf{w}^*$ by solving the following

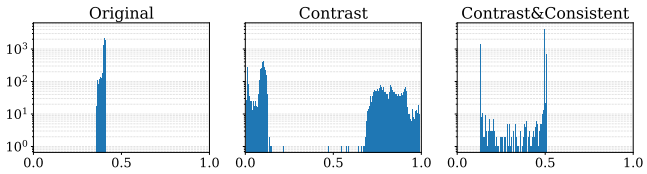


**Figure 1:** *Distribution of the facial mask weights learned by the various versions of CMAP.*

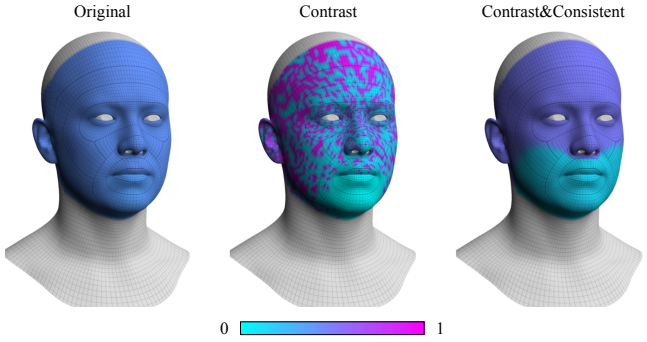


**Figure 2:** *Facial masks learned by the various versions of CMAP.*

**Table 1:** *Quantitative comparison of the CMAP variants.*

| region | $m_d \downarrow$ | $m_x \downarrow$ | $m_{\mathbf{AUC}} \uparrow$ |
|---|---|---|---|
| Original | $2.37 \pm 1.51$ | 9.50 | 54.94 |
| Contrast | $1.72 \pm 1.05$ | 6.21 | 65.57 |
| Contrast&Consistent | $\mathbf{1.51 \pm 1.00}$ | **5.43** | **69.66** |

minimization problem:

$$\begin{aligned}\mathbf{w}^* &= \arg\min_{\mathbf{w}} \ \alpha_{\text{data}}\mathcal{L}_{\text{data}} + \alpha_{\text{reg}}\mathcal{L}_{\text{reg}} \\ \mathcal{L}_{\text{data}} &= \|\mathcal{P}(\tilde{U}_s, \tilde{U}_t) - \tilde{U}_t\|_{\text{F}}^2 \\ \mathcal{L}_{\text{reg}} &= \max\left(0, \rho N - \|\mathbf{w}\|^2\right),\end{aligned}$$

where $\alpha_{\text{data}}, \alpha_{\text{reg}}$ are loss term weights, and $\rho$ is a hyperparameter set by the authors to 0.4.

We refer to this energy formulation as *Original*, and we found that optimizing this problem leads to a very narrow distribution of weights which do not clearly prefer some facial areas from others, as can be seen in Fig. 1 and Fig. 2. This further leads to suboptimal results, as shown in Tab. 1.

To encourage higher contrast in the learned weights, we add an additional energy term

$$\mathcal{L}_\sigma = -\sigma(\mathbf{w}),$$

where $\sigma$ computes standard deviation over a vector of values. This variant, which we refer to as *Contrast*, is forced to make a clear decision about which facial areas are relevant for the rigid alignment, as seen in Fig. 1 and Fig. 2. It is evident that the method tends to discard the jaw area, which is typically the least stable part across an expression set. While the results improve, as seen in Tab. 1, the mask appears noisy which harms the performance.

Therefore, we add an additional energy term encouraging local spatial consistency of the weights, defined as

$$\mathcal{L}_{\mathbb{N}} = \frac{1}{N}\sum_{i=1}^{N} \sigma(\mathbb{N}_k(\mathbf{w}_i)),$$

where $\mathbb{N}_k(\mathbf{w}_i)$ finds the weights of $k$ nearest neighbors of vertex $U_i$. Finally, we find the best weights as

$$\mathbf{w}^* = \arg\min_{\mathbf{w}} \ \alpha_{\text{data}}\mathcal{L}_{\text{data}} + \alpha_{\text{reg}}\mathcal{L}_{\text{reg}} + \alpha_\sigma\mathcal{L}_\sigma + \alpha_{\mathbb{N}}\mathcal{L}_{\mathbb{N}}.$$

We performed grid search over the loss term weights and number of neighbors $k$ on the validation set and eventually set them to $\alpha_{\text{data}} = 100, \alpha_{\text{reg}} = 0.01, \alpha_\sigma = 100, \alpha_{\mathbb{N}} = 100, k = 10$.

The final variant is referred to as *Contrast&Consistent*. As can be seen in Fig. 1, the distribution of the weights across the surface is not as extreme as in the case of *Contrast*, but it is still clearly bimodal and it discards the lower part of the face as shown in Fig. 2. This variant yields the best performance, as shown in Tab. 1 and thus we use it for all experiments in the main paper.

## References

[WSS18] WU C., SHIRATORI T., SHEIKH Y.: Deep incremental learning for efficient high-fidelity face tracking. *ACM Transactions on Graphics* (2018). 1