MoFA: Deep Convolutional Monocular Face Reconstruction

Paper: https://ieeexplore.ieee.org/document/8237663

Authors: Max-Planck-Institute & University of Luxembourg

▼ **Terminology**

Face reconstruction: estimating 3D model of face from images

Face reenactment: transferring expressions from one face to another

Monocular: from one image

1. Objective

To reconstruct a 3D face from an unsupervised single image using a CNN encoder network with generative decoder network.

---

2. CNN Encoder

1. Using existing encoder network (VGG-Face), we output a vector as below which defines face pose, shape, expression, skin reflectance and scene illumination.

The semantic code vector $\mathbf{x} \in \mathbb{R}^{257}$ parameterizes the facial expression $\boldsymbol{\delta} \in \mathbb{R}^{64}$, shape $\boldsymbol{\alpha} \in \mathbb{R}^{80}$, skin reflectance $\boldsymbol{\beta} \in \mathbb{R}^{80}$, camera rotation $\mathbf{T} \in SO(3)$ and translation $\mathbf{t} \in \mathbb{R}^3$, and the scene illumination $\boldsymbol{\gamma} \in \mathbb{R}^{27}$ in a unified manner:

$$\mathbf{x} = (\underbrace{\boldsymbol{\alpha},\ \boldsymbol{\delta},\ \boldsymbol{\beta}}_{\text{face}},\ \underbrace{\mathbf{T},\ \mathbf{t},\ \boldsymbol{\gamma}}_{\text{scene}}) \ . \tag{1}$$

2. Take the code vector from the encoder and generate an 3D reconstruction of the face from it):

- Represented the face as a triangle mesh of n=24,000 3d vertices set V (spatial embedding vector) and compute the pixels at each vertex using affine face model **(PCA)**:

$$\mathbf{V} = \hat{\mathbf{V}}(\boldsymbol{\alpha}, \boldsymbol{\delta}) = \mathbf{A}_s + \mathbf{E}_s\boldsymbol{\alpha} + \mathbf{E}_e\boldsymbol{\delta} \ .$$

- Average face shape computed based on 200 (100 male, 100 female) high-quality face scans

- V is the spatial embedding, i.e., the 3D coordinates of the mesh.
- $\hat{V}(\alpha, \delta)$ is the parameterized face model.
- A_s is the basic face shape.
- $E_s\alpha$ models the shape variation.
- $E_e\delta$ models the expression changes.
- α and δ are parameters controlling shape and expression, respectively.

3. Take the code vector from the encoder and generate a skin reflectance for each vertex:

$$\mathbf{R} = \hat{\mathbf{R}}(\boldsymbol{\beta}) = \mathbf{A}_r + \mathbf{E}_r\boldsymbol{\beta} \ .$$

---

3. Decoder (Non-NN)

Given the 3d scene reconstruction of vertices above, lets generate to a 2D image…

1. Compute where vertices go relative to camera position and orientation model using a parameterisation and mapping.
2. Represent colour and lighting of vertices using spherical harmonics (colour as a function of skin reflectance, vertex normal & neutral colour)

$$C(\mathbf{r}_i, \mathbf{n}_i, \boldsymbol{\gamma}) = \mathbf{r}_i \cdot \sum_{b=1}^{B^2} \boldsymbol{\gamma}_b \mathbf{H}_b(\mathbf{n}_i) \quad .$$

---

4. Loss Function

$$E_{\text{loss}}(\mathbf{x}) = \underbrace{w_{\text{land}} E_{\text{land}}(\mathbf{x}) + w_{\text{photo}} E_{\text{photo}}(\mathbf{x})}_{\text{data term}} + \underbrace{w_{\text{reg}} E_{\text{reg}}(\mathbf{x})}_{\text{regularizer}}$$

Loss consists of weighted:

- Landmark loss (optional): the loss around certain landmarks of the face… ie. nose

$$E_{\text{land}}(\mathbf{x}) = \sum_{j=1}^{46} c_j \cdot \left\| \mathbf{u}_{k_j}(\mathbf{x}) - \mathbf{s}_j \right\|_2^2$$

- Photo loss: the loss / difference over the entire photo between generated pixels and photo pixels

$$E_{\text{photo}}(\mathbf{x}) = \frac{1}{N} \sum_{i \in \mathcal{V}} \left\| \mathcal{I}\big(\mathbf{u}_i(\mathbf{x})\big) - \mathbf{c}_i(\mathbf{x}) \right\|_2$$

- Regularisation loss: enforces statistical plausibility by making sure parameters are close to average

$$E_{\text{reg}}(\mathbf{x}) = \sum_{k=1}^{80} \boldsymbol{\alpha}_k^2 + w_{\boldsymbol{\beta}} \sum_{k=1}^{80} \boldsymbol{\beta}_k^2 + w_{\boldsymbol{\delta}} \sum_{k=1}^{64} \boldsymbol{\delta}_k^2$$