Face Reconstruction and Expression Transfer for RGBD Images

Yimin Pan Yi Wei Weixiao Xia Xiyue Zhang

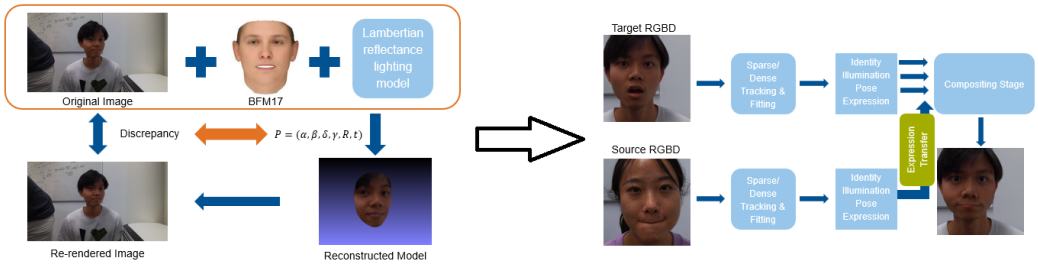


Figure 1. Method overview

Abstract

The aim of this project is to transfer the facial expressions from an actor in a source video to an actor in a target video. To achieve this, we first capture the facial performances of the source and target subjects using a RGB-D sensor. Then, we fit a parametric face model to a short controlled input sequence to jointly estimate identity and skin reflectance of both target and source subjects respectively. After that, for each frame of the target and source video, the estimated identities and skin reflectances will be used to estimate the expression of the actors and the scene lighting. For expression transfer, we compute the difference between source and it's neural expression in parameter space, and the offset is then added to the target parameters to match the source expressions. Finally, the synthesized target face is re-rendered into the target video stream in a photo-realistic manner.

1. Introduction

Actors rely on their facial expressions to accurately express emotions, and an appropriate facial expression could promote the development of the plot in a timely manner and consequently enhance the experience of the audience. Thus, we propose to improve the acting performance by improving the facial expression of the actor during the post-production, in a way that the desired facial expression will be transferred from a behind-the-scene veteran actor to the target actor.

In this project, we base our method on the work done by Justus et al. [8] with a slight variation. The biggest difference is that, Justus's method consists of an online method which runs at realtime thanks to it's custom GPU solver. However, we use ceres [1] as our solver due it's simplicity but the drawback is that the per-frame execution time is now about 30s.

Apart from Ceres another factor which makes our method offline is that we use a DL model [2] to detect facial landmarks as it can handle rotations with large degree (>30). But this model is by default implemented in the python's version of Pytorch so we had to write a python script apart to preprocess the frames in order to get the facial landmarks.

Furthermore, our method divides the fitting process into two stages. 1) We first reconstruct the shape identity, expression, and the face pose of the target face using a set of detected facial landmarks. 2) The parameters estimated in the first stage, together with skin reflectance and scene lighting, are jointly re/estimated in this stage using an *analysis-through-synthesis* approach.

2. Related work

Offline Reenactment Dale et al. [3] use a 3D multilinear model to achieve face replacement in video. The result is impressive but it does not allow to manipulate the facial expression of the target as it just replaces the face of the target actor to the source. The dubbing problem targeted by Garrido et al. in [4] is very similar to ours. For the foreign movies and TV productions, the original voice of an actor is replaced with a translation that is spoken by a dubbing actor in the country's own language. However, the sequence of phonemes and visems in the original and and the

dubbing language are different, the video-to-audio match is never perfect and is a discomfort for the audience. Garrido's 3DMM based approach alters the mouth motion of an actor in a video so that it matches the new audio track.

Online Reenactment Justus et al. [8] RGB-D based 3DMM fitting approach has achieved an impressive result in facial reenactment. Later on, Justus et al. has proposed a more generic and improved approach [9] which no more relies on depth data and synthesize the mouth regions exclusively from the target sequence.

3. Overview

Our approach use a linear parametric model for facial identity, expression, and albedo. The face model used in this project is BFM17 [5]. In addition to the face model, we use a lighting model with a Lambertian surface reflectance assumption to jointly estimate the environment lighting, in order to achieve a photorealistic re-rendering.

Face tracking and reconstruction We estimate the parameters of the statistical face model, the head pose, and the unknown incident illumination in the scene from the input depth and video data. We determine the model and lighting parameters by minimizing a non-linear least squares energy that measures the discrepancy between the RGB-D input data and the re-rendered output using the estimated face shape, pose, albedo, and lighting. We solve these unknowns in two steps: 1) shape identity, expression, and the face pose of the target face are estimated first using a set of detected facial landmarks. 2) The parameters estimated in the first stage, together with skin reflectance and scene lighting, are jointly estimated/refined in the second stage following the *analysis-through-synthesis* approach. We use Ceres to solve the above two non-linear problems.

Reenactment Once we have estimated the model parameters, the head pose and the lighting, the current expression of the source face is first subtracted from it's neutral expression, and then the offset is added to the neutral expression of the target face in order to transfer the expression. The synthesized target face with the source face's expression is then re-rendered and composited on top of the target video stream.

3.1. Parametric face model

Our parametric face model is defined by the following linear combinations:

$$M_{geo}(\alpha,\delta) = \boldsymbol{\mu}_{id} + \boldsymbol{U}_{id} diag(\boldsymbol{\sigma}_{id})\boldsymbol{\alpha} + \quad (1)$$
$$\boldsymbol{\mu}_{exp} + \boldsymbol{U}_{exp} diag(\boldsymbol{\sigma}_{exp})\boldsymbol{\delta}, \quad (2)$$

$$M_{alb}(\beta) = \boldsymbol{\mu}_{alb} + \boldsymbol{U}_{alb} diag(\boldsymbol{\mu}_{alb})\boldsymbol{\beta}, \quad (3)$$

where $\boldsymbol{\mu}_{id,exp,alb} \in \mathbb{R}^{3n}$ are the mean, $\boldsymbol{\sigma}_{id,exp,alb} \in \mathbb{R}^{m}$ the standard deviations and $\boldsymbol{U}_{id,exp,alb} = [\boldsymbol{u_1}, ..., \boldsymbol{u_n}] \in \mathbb{R}^{3nxm}$ are an orthonormal basis of principle components of shape, texture and expression.

In our experiment we use the first 100 principal directions to span the space of plausible facial shapes with respect to the geometric embedding, skin reflectance and facial expression.

3.2. Illumination model

We followed the original implementation in [8] assuming the lighting is distant and that the surfaces in the scene are predominantly Lambertian, which leads to use SH basis to simulate the incident illumination. Thus, for each channel of RGB, the irradiance of a vertex can be computed with its normal $\boldsymbol{n}$, scalar albedo $\boldsymbol{c}$ and 3-bands Spherical Harmonics(SH) basis according to [7]:

$$L(\gamma, \boldsymbol{n}, c) = c \cdot \sum_{k=1}^{b^2} \gamma_k y_k(\boldsymbol{n}), \quad (4)$$

with y_k being the k-th SH basis function and $\gamma = (\gamma_1, ..., \gamma_{b^2})$ the corresponding SH coefficients.

3.3. Image formation Model

In addition to the face and illumination models, we need a representation for the head pose and the camera projection onto the virtual plane. Let $\mathbf{S}$ denotes our image formation model and $\mathbf{P}$ denotes the parameters of this image formation model, we have that:

$$\mathbf{P} = (\boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{\delta}, \boldsymbol{\gamma}, \mathbf{R}, \mathbf{t})$$

where the first 4 parameters are related to the face and illumination model and the last 2 parameters controls the face pose. In total we have p = 100 + 100 + 100 + 27 + 3 + 3 = 333 parameters.

4. Model fitting

4.1. Input data

The input is provided by an RGB-D camera and consists of the measured input color sequence $C_{\mathcal{I}}$ and depth sequence $X_{\mathcal{I}}$. We will also compute a normal map $N_{\mathcal{I}}$ from the input depth sequence $X_{\mathcal{I}}$.

4.2. Implementation of the Image Formation Model

The image formation model $\mathcal{S}(\mathbf{P})$ is implemented by means of the GPU rasterization pipeline. Our GPU rasterizer is a cuda-parallelized version of a CPU rasterizer

described in [6]. The output of the rasterizer is the synthetic color $C_{\mathcal{S}}$, the 3D position $X_{\mathcal{S}}$, the pixel normal map $N_{\mathcal{S}}$, the vertex normal of all vertices that are used to render the current image, the barycentric coordinates of the each pixel and the indices of the vertices in the covering triangle.

As in [8], we only consider pixels belonging to the set $\mathcal{V}$ of pixels for which both the input and the synthetic data is valid.

4.3. Energy Formulation

The energy formulation of our approach is the same described in [8]:

$$E(\mathbf{P}) = E_{emb}(\mathbf{P}) + w_{col}E_{col}(\mathbf{P}) + w_{lan}E_{lan}(\mathbf{P}) + w_{reg}E_{reg}(\mathbf{P}) \tag{5}$$

Geometry Consistency Metric The geometry consistency is defined by the **point to point** and **point to plane** distance between the synthetic depth map and the input depth stream:

$$E_{emb}(\mathbf{P}) = w_{point}E_{point}(\mathbf{P}) + w_{plane}E_{plane}(\mathbf{P}) \tag{6}$$

where:

$$E_{point}(\mathbf{P}) = \sum_{\mathbf{p}\in\mathcal{V}} \|d_{point}(\mathbf{p})\|_2^2 \tag{7}$$

$$E_{plane}(\mathbf{P}) = \sum_{\mathbf{p}\in\mathcal{V}} d_{plane}^2(N_S(p), p) + d_{plane}^2(N_{\mathcal{I}}(p), p) \tag{8}$$

Color Consistency Metric The Color consistency is defined by the difference between the input RGB image and the rendered view:

$$E_{col}(\mathbf{P}) = \sum_{\mathbf{p}\in\mathcal{V}} \|\mathcal{C}_{\mathcal{S}}(\mathbf{p}) - \mathcal{C}_{\mathcal{I}}(\mathbf{p}\|_2^2 \tag{9}$$

Feature Similarity Metric The set of sparse facial landmars in the RGB stream used to compute the feature similarity is detected by a DL model described in [2]. It can precisely detect the set of facial landmarks even if the face is highly rotated ($\geq$ 30 degrees):

$$E_{lan}(\boldsymbol{P}) = \sum_{\boldsymbol{f}_j\in F} w_{conf,j}\|\boldsymbol{f}_i - \Pi(\Phi(\boldsymbol{v}_j))\|_2^2 \tag{10}$$

Regularization constraints The model parameters α, β, δ are regularized using the following energy:

$$E_{reg}(\mathbf{P}) = \sum_{i=1}^{100} \alpha_i^2 + \beta_i^2 + \delta_i^2 \tag{11}$$

Observe that we did not divide each term by its corresponding variance σ_i like in [8] because in our method the coefficients $\{\alpha_i, \beta_i, \delta_i\}$ is already explicitly multiplied by its corresponding variance σ_i.

4.4. Ceres solver

We use Ceres as our solver. As we have mentioned in the previous sections, our approach is slightly different than the approach presented in [8] as we divide the optimization pipeline into two stages.

1. In the first stage we estimate the identity, expression, and the head pose by optimizing the below non-linear energy:

   $$E(\mathbf{P}) = w_{lan}E_{lan}(\mathbf{P}) + w_{reg}E_{reg}(\mathbf{P}) \tag{12}$$

   Every landmark is considered as an residual block and are added to the Ceres Problem object.

2. In the second stage we refine the identity, expression, and estimate/refine skin albedo and scene lighting based on the parameters estimated in the previous stage by optimizing the non-linear energy described in 4.3 following the *analysis-through-synthesis* scheme.

   To optimize this problem, We consider every pixel $\mathbf{p} \in \mathcal{V}$ as an residual block together with the previous landmarks.

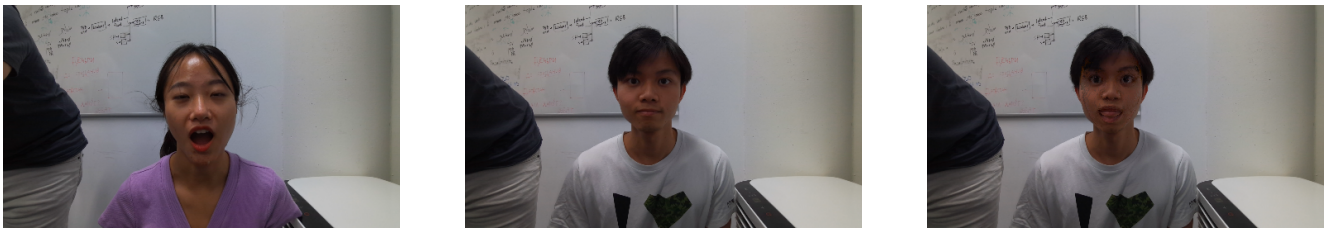
4.5. Initialization of Identity and Albedo

Just like in [8], in our approach the identity and skin reflectance are also preestimated on a short controlled sequence, and both remain constant in the rest of the frames.

But, unlike in [8] where an high resolution uv-map is computed to store the texture, here we directly project the vertices of the face model to the source image and takes the corresponding pixel color as the vertex's color.

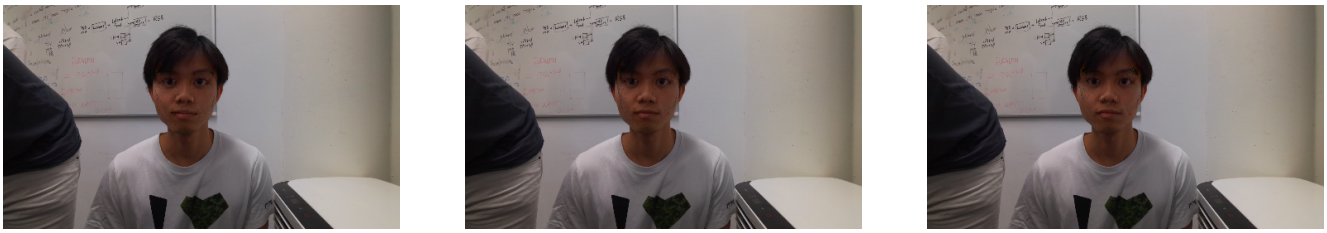
5. Results

We represent face reconstruction results from different images in the figure 4. The lighting condition and the facial texture are precisely captured.



(a) Source actor (b) Target actor (c) Composited output

Figure 2. Facial expression transfer. Observe that in this project we did not include a teeth model, so the final composition will look very weird if the mouth is open as the original color is not replaced.



(a) Frame t (b) frame t+1 (c) frame t+2

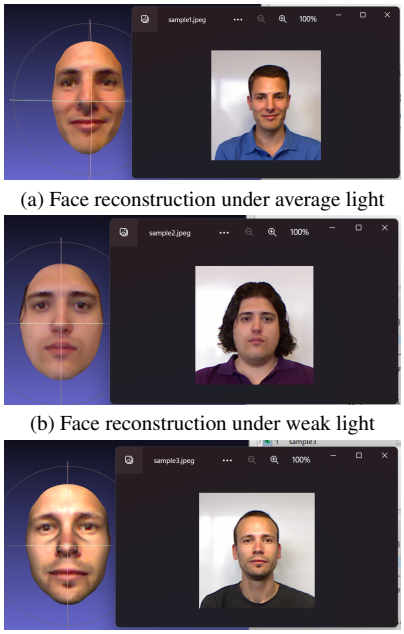
Figure 3. An example sequence.

The results for the facial expression transfer are represented in the figure 2. Although the face reconstruction result looks pretty impressive, the performance of the facial expression transfer is not satisfactory at all. The generated face mask is unstable and it produces noticeable position shiftments between frames even for static faces as shown in figure 3.

6. Conclusion

In this project we have implemented the face reconstruction and the facial expression transfer algorithm proposed in [8]. However, the reimplemented approach is far being real-time and has several drawbacks. So, we plan to do the following improvements in the future:

- Replace Ceres with the GPU solver described in the original approach.
- Find another reliable c++ facial landmark detector / Use TensorRT to speedup the current DL model.
- Extract texture from the source image to an uv-map in order to achieve a smoother and a more precise texture mapping.
- Increase the basis of the current blendshape model in order to produce finer expressions.
- Find the reason why the generated face mask is unstable between frames and solve it.
- Include a teeth model.



(a) Face reconstruction under average light

(b) Face reconstruction under weak light

(c) Face reconstruction strong light

Figure 4. Face reconstruction for faces under different lighting conditions.

References

[1] Sameer Agarwal and Keir Mierle. *Ceres Solver: Tutorial & Reference*. Google Inc. 1

[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 *International Conference on Computer Vision*, 2017. 1, 3

[3] Kevin Dale, Kalyan Sunkavalli, Micah K. Johnson, Daniel Vlasic, Wojciech Matusik, and Hanspeter Pfister. Video face replacement. *ACM Trans. Graph.*, 30(6):110, dec 2011. 1

[4] P. Garrido, L. Valgaerts, H. Sarmadi, I. Steiner, K. Varanasi, P. Pérez, and C. Theobalt. Vdub: Modifying face video of actors for plausible visual alignment to a dubbed audio track. *Comput. Graph. Forum*, 34(2):193204, may 2015. 1

[5] Thomas Gerig, Andreas Morel-Forster, Clemens Blumer, Bernhard Egger, Marcel Lthi, Sandro Schnborn, and Thomas Vetter. Morphable face models - an open framework, 2017. 2

[6] Patrik Huber, Guosheng Hu, Rafael Tena, Pouria Mortazavian, Willem Koppen, Matthias Rtsch, and Josef Kittler. A multiresolution 3d morphable face model and fitting framework. 02 2016. 3

[7] Ravi Ramamoorthi and Pat Hanrahan. A signal-processing framework for inverse rendering. In *Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '01, page 117128, New York, NY, USA, 2001. Association for Computing Machinery. 2

[8] Justus Thies, Michael Zollhöfer, Marc Stamminger, Christian Theobalt, and Matthias Nießner. Face2face: Real-time face capture and reenactment of RGB videos. *CoRR*, abs/2007.14808, 2020. 1, 2, 3, 4

[9] Justus Thies, Michael Zollhfer, Marc Stamminger, Christian Theobalt, and Matthias Niener. Face2face: Real-time face capture and reenactment of rgb videos. 2020. 2