Face Reconstruction Project

Gökçe Şengün, Dmitrii Pozdeev, Biray Sütçüoğlu, David Gichev
Technical University of Munich

1. Motivation and Idea

Accurate digital representation of facial features has significant potential in fields such as animation, gaming, and virtual reality, as well as in medical imaging, biometrics, and forensic science. This project focuses on facial reconstruction from a single RGB/RGB-D image, which is closer to the real-world scenario where only simple sensors or phones are available. Using a pre-trained PCA-based face model, we solve optimization tasks for face parameters by analysis by synthesis and obtain a complete 3D face model. We estimate the face's shape, expression, and albedo, optimizing spatial, photometric, and sparse terms. Firstly, we run image landmark detection using PipNet [2] to obtain facial landmarks and solve for sparse terms only. In the RGB-D setup, we add dense terms for better shape and expression approximation. Then, we fix the extrinsic, shape, and expression parameters and use photometric terms to restore texture. After solving the parameters, we apply our pipeline to face reenactment using the expression of the source actor to transfer it to the target actor model by copying expression parameters. Finally, we restore environment lightning by optimizing spherical basis coefficients. The code can be observed on the github page.

2. Related work

Our work is based on the method proposed in [5,6]. Using a similar pipeline, authors effectively implement optimization on CUDA, resulting in real-time transfer expression. The follow-up work extended the method to work with RGB-only images. Our implementation borrows the pipeline from the works mentioned above, but we focus on CPU instead and use the ceres library for cost function implementations.

3. Method

In this section, we present the overview of the whole pipeline and then subsequently specify details for every part. Given the face parametric model $\mathcal{M}(\mathbf{P})$ with parameters $\mathbf{P}$ and the person's RGDB-D image $\mathcal{I}$. The goal is to obtain parameters $\mathbf{P}'$, such that the projection of $\mathcal{M}(\mathbf{P}')$ is equal to the $\mathcal{I}$. We will use PCA-based morphable face models comprising pose $\mathbf{\Phi}$, shape $\boldsymbol{\alpha}$, albedo $\boldsymbol{\beta}$, illumination $\boldsymbol{\gamma}$, and expression $\boldsymbol{\delta}$ parameters. We consider only Lambertian surfaces and distant light sources. Thus, we are using 27 coefficients of the first three bands of spherical harmonics [4]. The model defines the mesh of n vertices V using pre-computed PCA basis:

$$M(\boldsymbol{\alpha}, \boldsymbol{\delta}) = \alpha_{mean} + E_{id}\boldsymbol{\alpha} + E_{exp}\boldsymbol{\delta} \quad (1)$$

$$M(\boldsymbol{\beta}) = \beta_{mean} + E_{alb}\boldsymbol{\beta} \quad (2)$$

where $E_{id} \in \mathbb{R}^{3n\times 199}$, $E_{expr} \in \mathbb{R}^{3n\times 100}$, $E_{alb} \in \mathbb{R}^{3n\times 199}$ are corresponding PCA basis for shape, expression and albedo; $\boldsymbol{\alpha}_{mean}, \beta_{mean}$ are average parameters of shape and albedo. The face reconstruction proceeds with analysis by synthesis fashion: at each step, we render the current face image through our model $\mathcal{M}(\mathbf{P})$, then we compare it with the actual image $\mathcal{I}$ and update parameters $\mathbf{P}$. Overall energy terms can be divided into three parts:

$$E = \lambda_{dense}E_{dense} + \lambda_{sparse}E_{sparse} + \lambda_{reg}E_{reg} \quad (3)$$

The dense term E_{dense} comprises of geometry term E_{geom} for computing the depths in the world space; and the color term E_{color}, which compares values with image $\mathcal{I}$ in the color space:

$$E_{dense}(\mathbf{P}) = \lambda_{geom}E_{geom} + \lambda_{color}E_{color} \quad (4)$$

$$E_{geom}(\mathbf{P}) = \lambda_{p2p}\sum_i \|\mathcal{M}(\mathbf{P})_i - D(\pi(\mathcal{M}(\mathbf{P})_i)\|^2 + \\ + \lambda_{p2plane}\sum_i \|[\mathcal{M}(\mathbf{P})_i - D(\pi(\mathcal{M}(\mathbf{P})_i)] \cdot n_i\|^2 \quad (5)$$

$$E_{color}(\mathbf{P}) = \sum_i \|\pi(\mathcal{M}(\mathbf{P})_i - \mathcal{I}(\pi(\mathcal{M}(\mathbf{P})_i))\|^2 \quad (6)$$

where the summation is over all the vertices and π is the projection on the image. We are computing photometric terms for every vertex instead of every pixel. For better quality, we are using the target color as an interpolation between colors of neighboring pixels projected onto the image vertex. Since we use a simple parametrization model for modeling complex material, skin, we also implemented

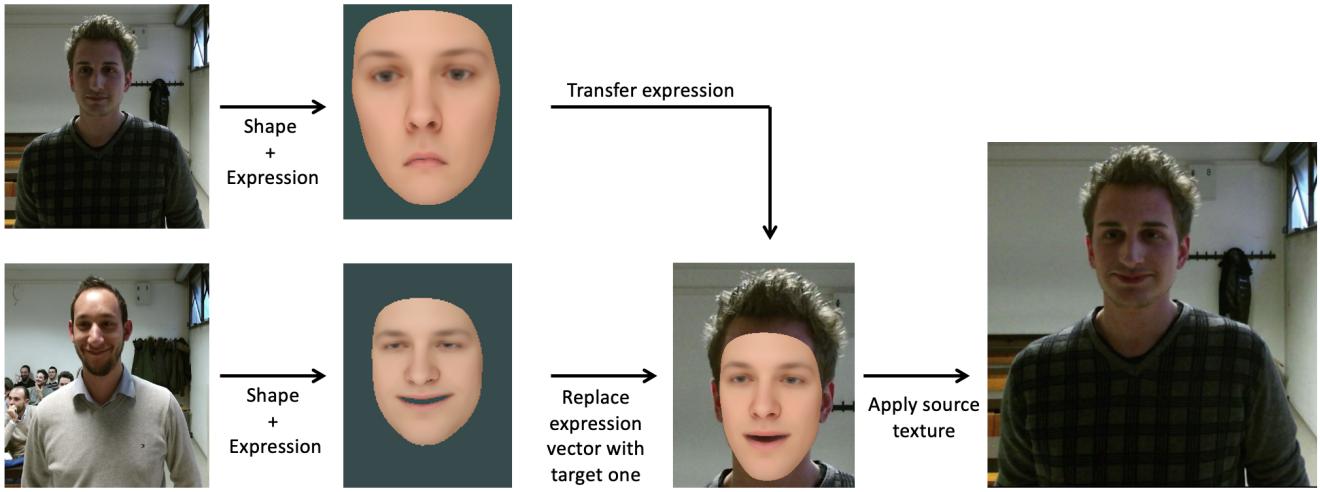

Figure 1. Transfer expression example. We find parameters independently using source (top left) and target (bottom left) actors. Then, we transfer expression coefficients from the source into the target. Finally, we project the source image into the mesh and visualize the mask above the source image.

mesh projection into the image. We have noticed that such projection can show good results even when the expression varies after fixing the texture of the vertices. Still, we observe apparent artifacts for large deviations of the shape and expression coefficients since the geometry is not perfectly restored.

Depth term We use point-to-point (p2p) and point-to-plane (p2plane) cost functions for depth alignment. We compute normals n_i for every vertex v_i by summing normals of the adjacent faces with weights. We use the area of the triangle as a weight for smoother results. However, since most dataset depth maps come with a lesser resolution than RGB images, we apply median and bilateral filters to smooth the normals of the point cloud. For more robust correspondences in the E_{geom}, we utilize standard techniques for pruning from ICP, pruning correspondence with large angles between normals (bigger than $60°$) and distances between corresponding points. After a single optimization step, we recompute normals, prune correspondence, and proceed until convergence. The dense term E_{dense} converges fast if meshes and point clouds are close enough. Therefore, we use the E_{sparse} term as a rough guide. For this, we are using facial landmarks, obtaining positions l_m on the image $\mathcal{I}$:

$$E_{sparse}(\mathbf{P}) = \sum_m \|\pi(\mathcal{M}(\mathbf{P}))_m - l_m\|^2 \qquad (7)$$

We have found that landmark correspondence of the face with mash landmarks is vital for stable optimization and a good initial point for extrinsic parameters. In the RGB-D setting, we run the Procrustes algorithm in 3D to obtain initial values for $\mathbf{R}, \mathbf{t}, \mathbf{s}$, by back-projecting landmarks using intrinsic matrix and depth map and force correspondence alignment between landmarks of the image and the mesh. We are using log-parametrization for $\mathbf{s}$, to avoid negative scaling. A sparse term is essential for convergence since the dense depth term is sensitive to the starting initialization point, and eradicating the sparse term results in the failure of our method. However, we noticed that using landmarks only in the Procrustes step - gives an appropriate initialization point for the depth term. Still, we are adding sparse terms for better convergence.

Since we are using a PCA-based model, coefficients are centered at 0, and we can restore their standard deviations from PCA singular values. To avoid overfitting, we also apply priors for parameter regularization. Omitting prior term results in unstable optimization and usually severe failures. We assume that coefficients are normally distributed, thus using squared loss:

$$E_{reg} = \sum_j \left(\frac{\boldsymbol{\alpha}_i}{\sigma_{id,j}}\right)^2 + \left(\frac{\beta_i}{\sigma_{alb,j}}\right)^2 + \sum_k \left(\frac{\delta_k}{\sigma_{exp,k}}\right)^2 \qquad (8)$$

RGB only We also apply our pipeline on the internet images, where only RGB is available. For this, we fix the intrinsic matrix using focal distance $50\,mm$ and image size on the device as $36 \times 25\,mm$. Then, we run sparse terms for shape, expression, and extrinsic parameters. Finally, we either solve for a photometric term or project the image into the mesh. Due to the absence of depth data, the mesh face focuses only on good projection; this may result in overfitted extrinsic and shape coefficients.

Lightning We use first three bands of sphrecial harmonisc to model lightning. For vertex v_i with normal n_i, we obtain color using Equation 9.

$$\mathcal{L}(\gamma, n_i, c) = c\left(\sum_{b=1}^{9} \gamma_b \cdot y_b(n_i)\right) \tag{9}$$

where y_b is the b spherical harmonic basis functions and γ_k are learned coefficients. After obtaining the lightning coefficients, we can restore the original face using β albedo coefficients. Also, we can utilize lightning for back-projection by diving color with the sum of the spherical harmonic.

3.1. Experimental setup

We use the Basel Face Model (BFM) [1] for our PCA-based model. We mainly focused on the 2017 no-mouth model for our experiments since it handles most cases and has fewer vertices. However, the 2019 model handles faces with open mouths better since it also provides the tongue. We optimize for 199 coefficients for shape and albedo, 100 for expression, and 7 for rotation, translation, and scale. For lightning, we have 27 coefficients, 9 per RGB channel. As for landmarks detection, we found PipNet with ResNet18 backbone sufficient for our purposes. We stick to the 300W annotation, consisting of 68 landmarks overall. We are using the open-source Ceres library for optimization, which supports custom-based cost functions. Specifically, we employ the Levenberg–Marquardt algorithm. We discovered that using all the vertices is unnecessary in the dense terms. Therefore, we only utilize half of the vertices to speed up computations.

4. Results

We demonstrate RGB-only results in Figure 2. The left column is the original image, while the central column is the learned mesh by the model. Finally, the last column shows mesh with texture, which was projected from the original image. Although RGB only setting can be used for transferring expression, we also utilize depth information for geometry alignment. Face reenactment results can be seen in Figure 1. Using accurate depth alignment, we relight obtained mesh in Figure 7. Further we discuss failure of the landmarks and RGB only in sections 5.1 and 5.3.

5. Analysis

In this section we present analysis of our pipeline.

5.1. Landmark detectors

Since landmark detection is vital for extrinsic initialization and the only source of the geometry in the RGB-only setting, we compared different landmark detection libraries. For this, we tried images with different ages, expressions, and camera angles. Subset of the photos can be observed in Figure 3. Overall, DLIB [3] demonstrates good results on

Figure 2. Results of the optimization pipeline using RGB only. In the center column, we apply a BFM model texture. The last column obtains texture from the projection mesh into the image.

centered images with the face looking at the camera; however, it fails with jaw detection on images with non-standard angles (see Figure 3). We noticed that PIPNet [2] detects accurate landmarks for eyes, eyebrows, and nose but often gives inaccurate predictions for the chin. We stick to PiPNet landmarks detection since it produces more stable results, especially on images with different camera angles.

5.2. Depth term

For the depth term, we are using both point-to-point and point-to-plane terms. As shown in Figure 5 depth term is essential for accurate alignment of the face with the target point cloud. Without depth, the face tends to fit into the projection-only quality, resulting in inexact face geometry. We have found it vital to use correspondence pruning since we are using squared loss, and outliers result in poor convergence. We tried both variants for the depth term, using p2p only and with the p2plane term. Omitting normals information leads to slower convergence (20 iterations) while using normals resulted in 3-4 steps.

For the target normals computations, we estimate nor-

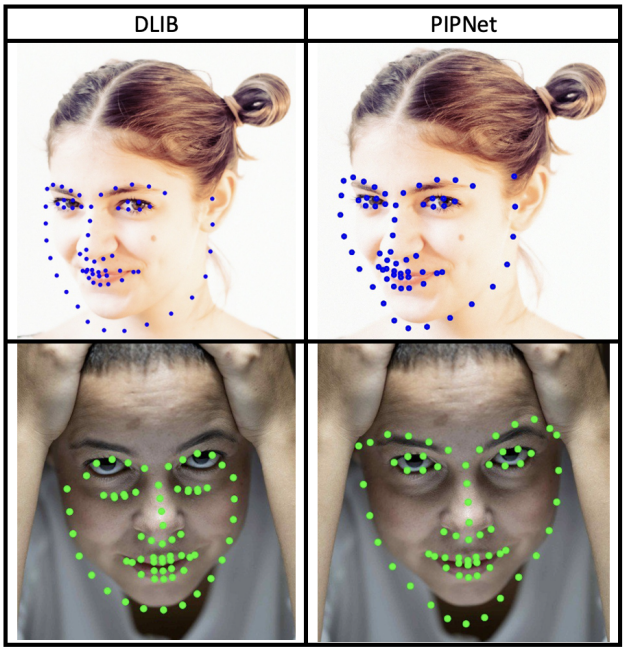

Figure 3. Comparison of DLIB and PiPNet landmark detectors. PipNet shows more robust behavior considering nonstandard camera angles.

mals by computing the tangent plane in every vertex, which can be obtained by gradient approximation in each point along horizontal and vertical directions. However, the normal map is uneven due to the noise of the depth sensor and the low resolution of the depth map. In the last image of the Figure 6, we apply median and bilateral filters to obtain smoother results.

5.3. RGB only

In real-world applications, depth data and camera intrinsics are unavailable; thus, we may rely only on sparse terms. Although sparse term results in good face projection, its size and location may be misleading, as in Figure 5. However, sparse terms can catch simple expressions, as shown in Figure 2. In this case, we found a balance between prior weights and sparse weight critical since we have only 68 target points to optimize for - the face model tends to overfit. Since we have insufficient information about actual depth, the sparse term fails to align the mesh with the face geometry, which results in misalignment with the original photo and pure projection of the image into the mesh. This phenomenon can be seen more clearly in the last column of Figure 2, taking a closer look at the lips region. The model struggles to cover geometry accurately because there are 12 lip landmarks close to each other. Nevertheless, resulting meshes can transfer expressions for simple images, as seen in the presentation video.

Figure 4. Failure of the transfer expression and image projection into the mesh. There is an edge of the lower lip from the original image, and the lips' angles look unnatural.

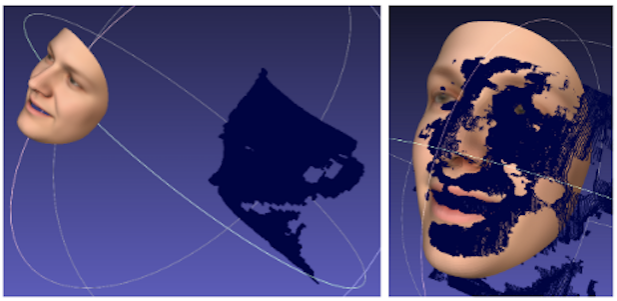

Figure 5. Visualization of the target point cloud and the restored mesh in meshlab using the sparse term only (first image) and sparse with depth term (second image). The dense depth term enables accurate alignment.

5.4. Transfer expression

We consider two settings for transferring expression. Given the learned model, we substitute expression coefficients with random noise in the first scenario, resulting in the face-to-change expression. We use face reenactment as the second scenario in Figure 1. Given the source and target actor, we obtain parameters for both images and then copy the expression from the source actor to the target model. We

Figure 6. Visualization of face normals, raw target normals from the depth map, and raw target normals processed with filters and mesh normals.

tried both settings in the RGB and RGB-D setup.

5.5. Color term

After fixing the geometry of the face, we optimize for texture coefficients. Since skin is a very complex material, the BFM texture can grasp only color tone, not considering the specks and hair of the target person. For this reason, we also apply ground truth color by the pure projection of every vertex into the image (see Figure 2). Although the projected image looks identical to the ground truth one, it produces visible artifacts when changing expression, as in Figure 4.

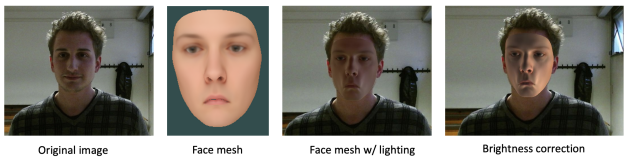

Figure 7. Mesh relighting by modifying SHs coefficients γ

In the RGB setting, we are using either texture from BFM or image projected into the mesh. Since we can obtain good alignment with the target point cloud in the RGB-D setting, we also demonstrate relighting results in Figure 7, which utilizes the normals of the vertices using Equation 9.

6. Conclusion

Using the PCA-based BFM model, we restore the full-face model using RGB and RGB-D inputs. We utilize sparse information using facial landmarks, prior information on coefficients and available depth maps for accurate geometry. Using learned parametric models, we transfer expression from source to target actor and modify face by adding random noise to expression coefficients. Finally, we consider the photometric term, solving texture coefficients or projecting the mesh into the image. For RGB-D inputs, our approach aligns 3D mesh with the original point cloud, enabling taking into account environment lightning. Finally, we can relight the original mesh by modifying Spherical Harmonic coefficients.

7. List of members

- Gökçe Şengün, (gokceesengun@gmail.com)
- Dmitrii Pozdev, (dmmpozdeev@gmail.com)
- Biray Sütçüoğlu, (biray.suetcueoglu@tum.de)
- David Gichev, (david.gicev@gmail.com)

References

[1] IEEE. *A 3D Face Model for Pose and Illumination Invariant Face Recognition*, Genova, Italy, 2009. 3

[2] Haibo Jin, Shengcai Liao, and Ling Shao. Pixel-in-pixel net: Towards efficient facial landmark detection in the wild. *International Journal of Computer Vision*, 129(12):3174–3194, Sept. 2021. 1, 3

[3] Davis E. King. Dlib-ml: A machine learning toolkit. *Journal of Machine Learning Research*, 10:1755–1758, 2009. 3

[4] Claus Müller. *Spherical Harmonics*. Springer Berlin Heidelberg, 1966. 1

[5] Justus Thies, Michael Zollhöfer, Matthias Nießner, Levi Valgaerts, Marc Stamminger, and Christian Theobalt. Real-time expression transfer for facial reenactment. 34(6), nov 2015. 1

[6] Justus Thies, Michael Zollhöfer, Marc Stamminger, Christian Theobalt, and Matthias Nießner. Face2face: Real-time face capture and reenactment of rgb videos, Jul 2020. 1