Face Reconstruction

Leo Keber, Fraj Yassine Lakhal, Neli Shahapuni
Technical University of Munich (TUM)
leo.keber@tum.de, fyassine.lakhal@tum.de, neli.shahapuni@tum.de

Abstract

This project aims to reconstruct a face from RGB-D monocular video data recorded using Intel RealSense. The input is mapped onto the Basel Face Model (BFM), a 3D morphable model, and optimized for pose, shape, expression, albedo, and illumination. The input and BFM are initially aligned in vertex space using Procrustes with the 68 facial landmarks. Sparse optimization is then applied to estimate an initial shape and expression, followed by dense optimization across all vertices. Dense geometry optimizes for shape and expression using point-to-point and point-to-plane constraints, while color optimization gives albedo and illumination. Through iterative Energy Minimization (Sparse Terms, Dense Terms, and Regularization), the Basel Face Model morphs into the input face. For each 2D image, the output is a 2D image overlaid with the morphed face.

1. Introduction

Our motivation for pursuing the Face Reconstruction project is to transfer a face from an input image onto a 3D parametric face model. This is an important step in the overall task of facial re-enactment, that is, transferring the face of a source actor to a target actor. The scope of this project is to reconstruct any human face by optimizing for various parameters, namely pose, shape, expression, albedo, and illumination. We achieve this by using the Basel Face Model (BFM) as our 3D morphable model. Furthering this project has practical applications, such as dubbing actors or characters, ensuring that a wider audience can enjoy content in their native languages.

2. Related Work

2.1. Face2Face

Thies et al. achieve real-time facial reenactment of a monocular target video sequence, where the expression of the source actor is transferred onto a target actor. They use statistical PCA-based face model for identity, expressions, and skin reflectance (albedo). Additionally, instead of copying the source mouth or using a generic 3D model, they use the best-matching mouth images from the target for natural-looking results. Their research achieves the first real-time facial reenactment method that works with monocular RGB video only. Ultimately, this research opens up new possibilities for VR/AR, teleconferencing, and real-time video dubbing. [8]

2.2. Real-time Expression Transfer for Facial Reenactment

Thies et al. develop a method for real-time facial reenactment, transferring expressions between source and target actor. They introduce a novel analysis-through-synthesis approach for face tracking, which maximizes photometric consistency between the input and re-rendered output video. With the use of a GPU-based Gauss-Newton solver, they achieve real-time optimization. The result is successful reenactment even under varied lighting conditions and head movements. They also preserve wrinkle-level detail transfer from source to target. Future research could improve on illumination modeling, handling extreme expressions, and identity-specific expression mapping. [7]

2.3. State of the Art on Monocular 3D Face Reconstruction, Tracking, and Applications

This paper summarizes the recent developments in monocular facial performance and their applications, such as performance-based animation and real-time reenactment. The authors reviewed optimization-based reconstruction algorithms that recover and track 3D facial models from single-camera RGB and RGB-D input. Their analysis covered key concepts of image formation, common assumptions, and simplifications, as well as various priors used to improve the reconstruction process. Lastly, they also discussed optimization techniques for obtaining dense, photo-geometric 3D face models and explored applications in motion capture, facial animation, and video editing. [9]

3. Project Setup

3.1. Basel Face Model

The Basel Face Model (BFM) [4] is a statistical 3D face model based on Principal Component Analysis (PCA), capturing variations in shape, expression and color (albedo). The model is stored in an HDF5 file, which contains:

- **Mean vectors:** We load the mean shape, expression, and color vectors to serve as the baseline for generating the face.
- **PCA Components:** We also extract the principal components for shape, expression and color (albedo) to model the variations.
- **Variances:** We further need variances associated with each principle component to scale the PCA coefficients.

3.1.1 Parametric Model Initialization

Following [4], we use the initialized model parameters for the 3D face geometry and 3D face albedo as:

$$\mathcal{M}_{\text{geo}} = \boldsymbol{\mu}_{\text{id}} + E_{\text{id}}\boldsymbol{\alpha} + E_{\text{exp}}\boldsymbol{\delta}, \quad \mathcal{M}_{\text{alb}} = \boldsymbol{\mu}_{\text{alb}} + E_{\text{alb}}\boldsymbol{\beta}$$

$\boldsymbol{\mu}_{\text{id}}$ being the mean identity (shape) and $\boldsymbol{\mu}_{\text{alb}}$ the mean albedo (color) and E_{id}, E_{exp} and E_{alb} being the principal components for identity, expression and albedo respectively. The parameters $\boldsymbol{\alpha}, \boldsymbol{\delta}, \boldsymbol{\beta}$, the coefficients for identity, expression and albedo, are optimized to minimize the energy function $E(\boldsymbol{\mathcal{P}})$, which combines landmark alignment, photometric consistency, and geometric regularization.

3.1.2 BFM Landmarks

The BFM provides 40 predefined landmarks, which do not correspond to the 68 landmarks used by dlib. To address this, we manually picked a custom set of landmarks inside Meshlab and we store them in a txt file. The landmarks cover key facial features such as the eyes, nose, lips, and eyebrows. The carefully selected landmarks represent anatomically significant points on the face, enabling accurate alignment through the use of Procrustes.

3.1.3 Face Mesh and Normals

The BFM includes a set of triangular faces, which defines the connectivity of the vertices to form a 3D mesh. Additionally, the model computes vertex normals to facilitate rendering and shading. The normals are updated by averaging the face normals of adjacent triangles, ensuring smooth shading across the mesh.

3.2. Data Acquisition and Preprocessing

Our search for a dataset that provides 3D data that does the combining of both RGB images and corresponding depth information—along with comprehensive ground truth landmark annotations was unsuccessful. While the Labeled Faces in the Wild (LFW) dataset [5] is one of the most widely used in the face recognition community, it lacks depth data. Although some variations of LFW include depth information, they do not offer the 68 landmark annotations required for our pipeline. Therefore, we use an rgb-d camera that will be our input data source.

3.2.1 Video Data Acquisition

The Intel RealSense D415 camera [1] to captures synchronized color and depth data. The camera records video streams in .bag files, which contain both RGB and depth information. We process the .bag file to extract individual frames. Each frame contains:

- **Color Data:** RGB values for each pixel, representing the visual appearance of the face.
- **Depth Data:** Per-pixel depth values, which provide the distance from the camera to the surface of the face.
- **Camera Parameters:** We extract the intrinsic and extrinsic matrices from the camera's metadata. The intrinsic matrix includes focal lengths and optical centers, while the extrinsic matrix defines the camera's position and orientation in 3D space.
- **Resolution:** The width and height of the captured frames are determined from the camera's intrinsics.

We process the extracted frames sequentially. For each frame, we align the depth and color data to ensure spatial correspondence.

3.2.2 Landmark Detection

We detect the facial landmarks using a pre-trained model (e.g., Dlib's 68-point facial landmark detector) [6]. These landmarks are used to guide the 3D reconstruction process. We project the 2D landmarks into 3D space using the depth data and camera parameters, providing a sparse 3D representation of the face.

3.2.3 Storing of Input Data

To optimize our loading process, we extract color, depth frames, camera intrinsics and extrinsics plus the resolution from the bag file. We serialize this input data using the cereal library and store them into a JSON file. The serialized file contains all the necessary data for reconstruction which we can load in subsequent runs, eliminating the overhead of reprocessing the bag file.

4. Technical Approach

4.1. Initial Alignment

We initially align the Basel Face Model with our input frame, to ensure it is approximately positioned, oriented and scaled for the following optimization. To do this, we apply a Procrustes analysis, consisting of three steps: rotation, scaling and translation. We use the detected landmarks of the input frame trg_i with mean μ_{trg} and the landmarks of the Basel Face Model src_i with mean μ_{src}.

To align the orientation of the Basel Face Model with that of the face in the input frame, we first estimate the rotation matrix R. This is done by computing the cross-covariance matrix between our two sets of landmarks:

$$A = Target^T \cdot Source, \tag{1}$$

where $Target$ corresponds to the landmarks of our input frame, and $Source$ to the landmarks of the Basel Face Model. We then perform Singular Value Decomposition (SVD) on A. To get the rotation we take the left (U) and right (V) singular vectors from the Singular Value Decomposition (SVD) of A, and use a corrective matrix $D = U \cdot V^T$.

$$R = U \cdot D \cdot V^T \tag{2}$$

Next, we compute the scale S based on the relative magnitudes of the landmarks of the Basel Face Model and the landmarks of the input frame. The resulting scale factor ensures that the Basel Face Model is proportionally resized to fit the size of the face in the input frame.

$$S = \sqrt{\frac{\sum_{i=1}^{n} ||trg_i - \mu_{trg}||^2}{\sum_{i=1}^{n} ||src_i - \mu_{src}||^2}} \tag{3}$$

After that, we compute the translation T based on the difference between the mean of the detected landmarks of the input frame and the mean of the rotated and scaled landmarks of the Basel Face Model.

$$T = \mu_{trg} - S \cdot R \cdot \mu_{src} \tag{4}$$

These three steps result in a transformation matrix M.

$$M = \begin{bmatrix} S \cdot R & T \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} s_x R_{xx} & s_x R_{xy} & s_x R_{xz} & t_x \\ s_y R_{yx} & s_y R_{yy} & s_y R_{yz} & t_y \\ s_z R_{zx} & s_z R_{zy} & s_z R_{zz} & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{5}$$

This transformation matrix is later applied to each vertex of the Basel Face Model.

4.2. Optimization

We reconstruct the unknown optimal parameters P^* by minimizing an energy function $E(P)$, which consists of data and prior terms.

$$E(P) = \underbrace{E_{dense}(P) + E_{sparse}(P)}_{data} + \underbrace{E_{reg}(P)}_{prior} \tag{6}$$

The data term uses sparse and dense terms to enforce constraints on vertex and pixel level [9]. The prior term ensures that the reconstruction is still plausible [8]. The optimization problem is formulated as a non-linear least squares problem:

$$P^* = argmin_P E(P) \tag{7}$$

Due to the high-dimensionality of this problem, we need a second order solver. We use the by ceres [2] provided implementation of the Levenberg-Marquardt algorithm. The parametric face model is optimized iteratively, updating shape, expression and color to best match the input.

4.2.1 Sparse Optimization

For sparse optimization, we minimize the distance between the already in world space projected landmarks l_i, detected in the input frame, and the corresponding landmarks of the parametric face model $M(P)$. This alignment can be formulated as the following energy function:

$$E_{sparse}(P) = \Sigma_i ||M_i(P)_{landmark} - l_i||_2^2 \tag{8}$$

Using sparse landmark constraints is very important due to the non-linearity of the energy minimization [9]. The initial fitting of the face brings the vertices of the parametric face model closer to their target points, which leads to faster convergence for the dense terms.

Due to potential outliers in the detected landmarks for the jawline, we chose to omit jaw landmarks entirely, even though this might result in a less accurate fit. Since we only have 68 landmarks, even a few outliers, because of inaccuracies in dlib's landmark detection or misinterpretations, for example due to hair, can lead to incorrect depth values and therefore impact the optimization. Because this issues do not occur that often for landmarks corresponding to the eyes, eyebrows, nose or mouth, we see it as sufficient to only omit the landmarks for the jaw. This leaves us with 40 remaining landmarks.

4.2.2 Dense Optimization

After the Basel Face Model is initially fitted using sparse optimization, we further refine it through dense optimization. Therefore we are using two terms: A dense color

term, which minimizes the differences in albedo between the Basel Face Model and the input frame. And a dense depth term, which aligns the facial geometry with the depth data of the input frame.

$$E_{dense}(P) = E_{geom}(P) + E_{color}(P) \tag{9}$$

For the dense color term, we minimize the difference between the RGB values of each vertex in the parametric face model $M(P)$ and the correspondending pixel in the input frame. Each vertex is projected in image space, and its color value is compared to the observed pixel data at that position:

$$E_{col}(P) = \sum_i ||M_i(P)_{color} - I(\pi(M_i(P)))||_2^2 \tag{10}$$

The dense depth term ensures that the reconstructed face model aligns with the observed depth data. Here, for each vertex, its correspondent pixel of the input frame is projected into world space using its depth value. The optimization then minimizes differences between the vertices of the Basel Face Model and their corresponding depth observations. Since the sparse optimization has already aligned the model roughly, we assume that the initial fit is reasonable. To avoid outliers, we discard vertex correspondences with large distance deviations, as they are probably from non-facial regions.

The depth term consists of two components:

- A point-to-point term that minimizes the distance between model and observed points.
- A point-to-plane term that uses surface normals to improve alignment.

$$E_{geom}(P) = \underbrace{\sum_i \left\| M_i(P) - D(\pi(M_i(P))) \right\|_2^2}_{\textit{Point-to-point}} \tag{11}$$

$$+ \underbrace{\left[\big(M_i(P) - D(\pi(M_i(P)))\big) \cdot n_i \right]^2}_{\textit{Point-to-plane}} \tag{12}$$

The dense color and depth terms are combined for joint optimization. This way we keep both photometric and geometric consistency throughout the fitting process.

4.2.3 Regularization

To prevent degeneration of facial geometry and color, we use a statistical regularization term on our model parameters. Therefore, we divide every parameter by its corresponding standard deviation. This ensures that the estimated parameters stay close to the mean [9]. The regularization energy is formulated as follows:

$$\begin{aligned} E(P)_{reg} = &\sum_{i=1}^{n_{id}} \omega_{id} \cdot \left(\frac{\alpha_i}{\sigma_{id,i}} \right)^2 \\ &+ \sum_{i=1}^{n_{exp}} \omega_{exp} \cdot \left(\frac{\beta_i}{\sigma_{exp,i}} \right)^2 \\ &+ \sum_{i=1}^{n_{alb}} \omega_{alb} \cdot \left(\frac{\delta_i}{\sigma_{alb,i}} \right)^2 \end{aligned} \tag{13}$$

To allow individual control over shape (α), expression (δ) and color (β), we add a weight ω to each term.

4.3. Further Improvements

4.3.1 Subsampling

To optimize efficiency in dense optimization, we apply random subsampling of vertices per iteration. Each iteration selects a subset of points from the 27,000+ vertices using random shuffling and limits processing to a predefined maximum number of samples. This approach balances computational cost and accuracy by ensuring that different regions contribute to the optimization while preventing redundant residuals. Additionally, outlier vertices—based on depth differences—are automatically filtered out, improving overall convergence speed.

4.3.2 Color Blending

We as well integrate color blending by aligning albedo and illumination across frames. With this, we reconstruct facial color while preserving realistic texture.

4.3.3 Texture Mapping

We then integrate Texture mapping into our model by comparing the colors extracted from the input data with the results obtained after applying the color optimization process. This approach directly maps pixel colors onto the reconstructed model, preserving the original image details.

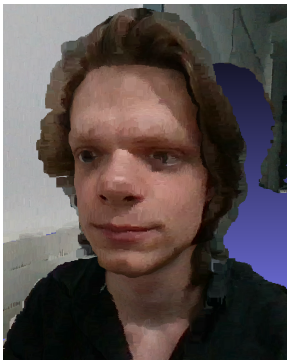
4.3.4 Laplacian Smoothing

We then apply Laplacian smoothing to the 3D mesh to reduce noise, in order to have a smoother surface while preserving the overall shape. The small irregularities in the final mesh arise from inaccurate depth data and optimization artifacts. We apply this transformation by iteratively adjusting each vertex position towards the average of its neighboring vertices using a smoothing factor lambda.

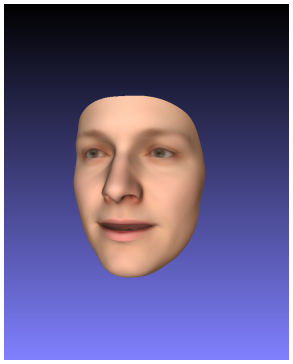
5. Evaluation

5.1. Results

Here you can see the optimization of the Basel Face Model step-by-step. These images were taken in Meshlab [3].



(a) Backprojected Image



(b) BFM after Procrustes

Figure 1. Initial Data

(a) BFM after sparse optimization

(b) BFM after dense optimization (without color)

Figure 2. Optimized Geometry

To further visualize our results, we rendered the Basel Face Model on top of our input frame.

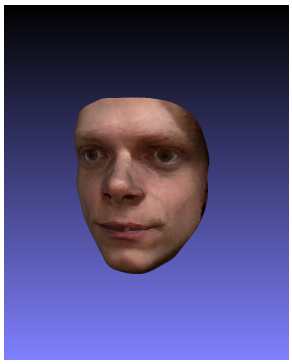
5.2. Performance Analysis

5.3. Accuracy

To evaluate how accurate our results are, we plot the geometric and photometric error for every step of our approach.

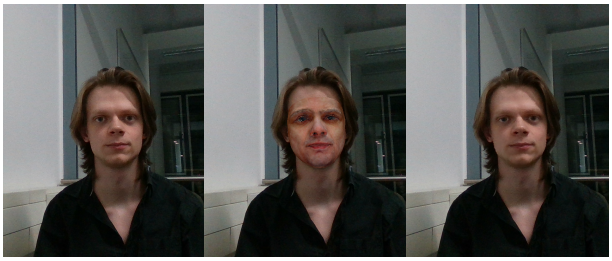
As expected is the geometric error quite high after Procrustes. After sparse optimization the error is lower around the landmarks. Due to the removal of the landmarks corresponding to the jaw, the error for the outer part of the face is very high. After dense optimization, the error decreases significantly, but we can still notice that the jaw is not perfectly aligned. For the photometric error we can see that after the color optimization and especially after the texture mapping, the error is significantly lower.

(a) BFM after dense optimization



(b) Mapped texture

Figure 3. Optimized Color



(a) Input Frame (b) Dense optimization (c) Mapped texture

Figure 4. Rendering of final results

(a) Input Frame (b) Dense optimization (c) Mapped texture

Figure 5. Rendering of final results

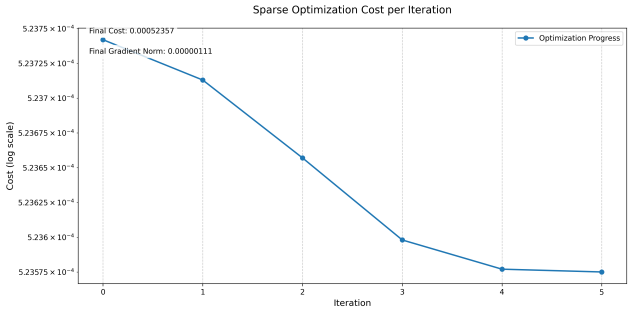


Figure 6. Sparse Optimization Loss

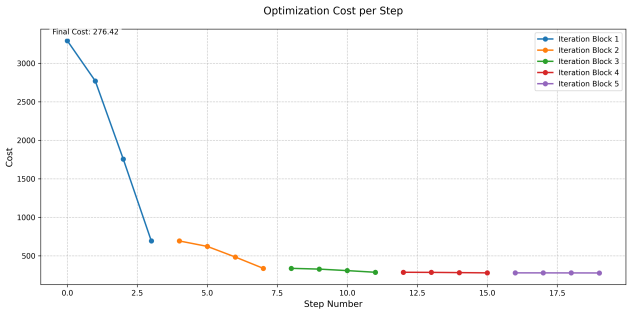


Figure 7. Dense Optimization Loss

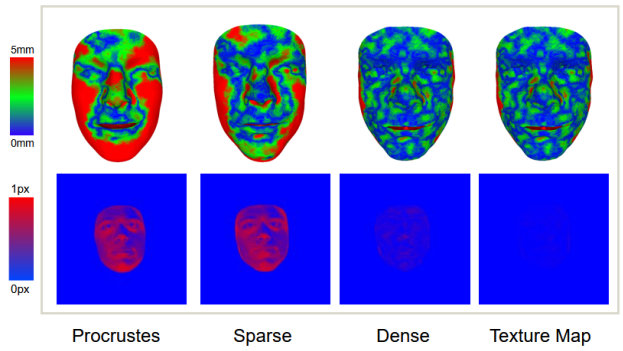


Figure 8. Geometric and photometric error

We also plot the loss curves for sparse and dense optimization separately.

6. Future Work

6.1. Teeth Proxy

The current face reconstruction model lacks explicit teeth representation, leading to uneven results in open-mouth expressions. Future improvements include:

- Integrate a parametric teeth model into the Basel Face Model for more realistic facial reconstructions.
- Use marker-based tracking to improve the positioning and alignment of inner-mouth structures.

6.2. Expression Transfer

Integrate correctly an expression transfer function to take the expression from a source face and transfer it onto a target face using landmark-based transformations. Future improvements include:

- Enhancing landmark correspondences to improve the accuracy of transferred expressions.
- Automating the process: The current method requires manual selection of input faces. A future improvement could automate the selection and alignment of source and target faces based on similarity metrics.

References

[1] Intel® realsense™ depth camera d415. https://www.intelrealsense.com/depth-camera-d415/. Accessed: 2025-02-14. 2

[2] Sameer Agarwal, Keir Mierle, and The Ceres Solver Team. Ceres Solver, 10 2023. 3

[3] Paolo Cignoni, Marco Callieri, Massimiliano Corsini, Matteo Dellepiane, Fabio Ganovelli, and Guido Ranzuglia. MeshLab: an Open-Source Mesh Processing Tool. In Vittorio Scarano, Rosario De Chiara, and Ugo Erra, editors, *Eurographics Italian Chapter Conference*. The Eurographics Association, 2008. 5

[4] Thomas Gerig, Andreas Morel-Forster, Clemens Blumer, Bernhard Egger, Marcel Lüthi, Sandro Schönborn, and Thomas Vetter. Morphable face models - an open framework. In *something*, 2017. 2

[5] Gary B. Huang, Manu Ramesh, Tamara Berg, and Erik Learned-Miller. Labeled faces in the wild: A database for studying face recognition in unconstrained environments. Technical report, University of Massachusetts, Amherst, 2007. 2

[6] Davis E. King. Dlib-ml: A machine learning toolkit. *Journal of Machine Learning Research*, 10:1755–1758, 2009. 2

[7] Justus Thies, Michael Zollhöfer, Matthias Nießner, Levi Valgaerts, Marc Stamminger, and Christian Theobalt. Real-time expression transfer for facial reenactment. *ACM Trans. Graph.*, 34(6), Nov. 2015. 1

[8] J. Thies, M. Zollhöfer, M. Stamminger, C. Theobalt, and M. Nießner. Face2face: Real-time face capture and reenactment of rgb videos. In *Proc. Computer Vision and Pattern Recognition (CVPR), IEEE*, 2016. 1, 3

[9] M. Zollhöfer, J. Thies, P. Garrido, D. Bradley, T. Beeler, P. Pérez, M. Stamminger, M. Nießner, and C. Theobalt. State of the art on monocular 3d face reconstruction, tracking, and applications. *Computer Graphics Forum*, 37(2):523–550, 2018. 1, 3, 4