3D Face Reconstruction from a Single Image using Parametric Model Fitting

Ana Carolina Ramos Cunha
03811403

Arjun Bommadevara
03811730

Bilgehan Savgu
03808347

Hürkan Uğur
03810239

Nils Collins
03811602

Abstract

*We present a practical pipeline for reconstructing a textured 3D face mesh from a single RGB image by fitting a parametric face model such as FLAME or BFM. Our method jointly estimates head pose, shape, and facial expression coefficients from detected 2D landmarks, using a precomputed semantic-to-mesh correspondence derived directly from the model's 3DMM parameters. The process begins with model inspection to extract and verify landmark indices from the shape representation. Given an input image, 68 facial landmarks are detected and matched to their 3D counterparts, enabling an initial pose estimation via a Perspective-*n*-Point (PnP) approach. We then perform iterative optimization over pose, shape, and expression parameters to minimize the reprojection error, with early stopping based on convergence criteria. Finally, the fitted geometry is textured either from the input image (FLAME) or by fitting a statistical albedo model (BFM), producing a realistic and identity-preserving mesh ready for rendering. This modular approach ensures robustness, interpretability, and reproducibility in single-image 3D face reconstruction.*

1. Introduction

Reconstructing a 3D face from a single RGB image is challenging due to the inherent ambiguity of recovering depth and shape from a 2D projection. Parametric 3D Morphable Models (3DMMs) [1], such as the Basel Face Model (BFM) [4] and FLAME [2], address this by representing shape, expression, and albedo with a compact set of learned parameters. Our goal is to estimate these parameters by fitting the model to an image in a controlled, verifiable manner.

Instead of a single monolithic optimization, we adopt a staged pipeline. We first inspect the 3DMM to extract a semantic-to-mesh landmark mapping, ensuring precise correspondence between detected 2D facial landmarks and 3D vertices. Pose is initialized via a Perspective-*n*-Point (PnP) formulation and then refined jointly with shape and expression parameters by minimizing the reprojection error. Depending on the model, texture is obtained either from the input image (FLAME) or by fitting a statistical albedo model (BFM).

Our contribution is a robust, modular framework that cleanly separates model inspection, landmark detection, and parameter fitting, providing a reliable foundation for full single-image 3D face reconstruction.

2. Technical Approach

Our system is designed as a sequential pipeline, illustrated in Fig. 1. The full pipeline aims to minimize a total energy function combining sparse, dense, and regularization terms:

$$E_{total} = E_{sparse} + E_{dense} + E_{reg} \tag{1}$$

In this paper, we focus on the initial pose estimation, which is driven by the sparse landmark alignment term, E_{sparse}. We utilize the Basel Face Model [4] as our 3D prior, with the potential to extend the framework to other models like FLAME [2].

2.1. 3D Model and Parameterization

We begin by loading the BFM from its HDF5 representation. The model provides a mean face shape ($\bar{S}$), a mean color vector ($\bar{C}$), and principal component analysis (PCA) bases for shape, color, and expression. A specific facial instance is synthesized from a set of weights (w) according to the formula:

$$\text{Instance} = \text{Mean} + (\text{PCABasis} \cdot w) \tag{2}$$

This generative capability forms the core of our "analysis-by-synthesis" approach. For the initial pose fitting stage detailed in this paper, we simplify this by using

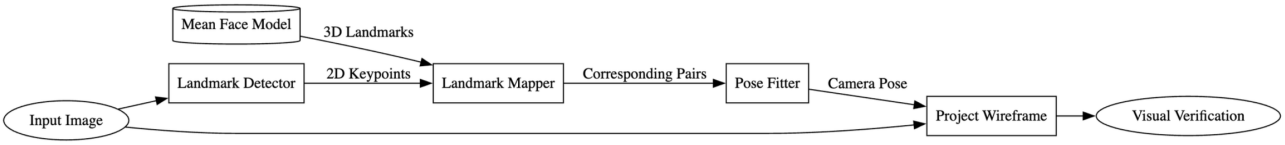

Figure 1. The proposed pipeline for 3D face pose estimation. An input image is processed by a Landmark Detector to find 2D keypoints. The Landmark Mapper establishes a correspondence between these 2D points and known 3D landmarks on a mean face model. The Pose Fitter uses these corresponding pairs to solve for the camera pose (rotation, scale, translation). Finally, the solved pose is used to project the full 3D wireframe onto the original image for visual verification.

only the mean shape ($\bar{S}$) as a rigid 3D template, effectively setting all shape and expression weights to zero.

Verification. The model components are validated by exporting the mean shape, mean shape with per-vertex color, and mean shape with the mean expression offset applied as separate '.obj' files for inspection in 3D software.

2.2. 2D Landmark Detection

Given an input image I, we employ a landmark detector module, which serves as a wrapper for a pre-trained library. We use Dlib's face detector and shape predictor [3] to localize a set of 68 specific 2D facial landmarks, $\{p_{2D}\} \in \mathbb{R}^2$. This module provides the 2D target points for our fitting process.

Verification. The accuracy of this stage is confirmed by generating a debug image where the 68 detected points are drawn on the input image, allowing for immediate visual inspection of the detector's performance.

2.3. Semantic Landmark Correspondence

A crucial step is to establish a correspondence between the 68 landmarks from Dlib and the semantically meaningful landmarks defined on our 3D model. These 3D landmark locations are loaded from the metadata within the HDF5 model file. Our LandmarkMapper module creates this bridge by defining a static map between Dlib indices and their corresponding semantic names in the 3D model's definition (e.g., Dlib index 30 maps to "center.nose.tip").

This produces two corresponding point lists of size K:

- A list of 3D object points, $\{P_{3D}\} \in \mathbb{R}^3$, from the mean face.
- A list of 2D image points, $\{p_{2D}\} \in \mathbb{R}^2$, from Dlib's output.

Verification. To validate the mapping, we generate a debug image where each mapped 2D landmark is tagged with its semantic name, providing unambiguous proof of correct correspondence.

2.4. Two-Stage Pose Fitting

With the corresponding 2D/3D point sets, we solve for the pose of a virtual camera. This is a classic Perspective-n-Point (PnP) problem. We use a weak perspective camera model:

$$p_i = s \cdot \Pi \cdot (R \cdot P_i) + t_{2D} \tag{3}$$

where R is the rotation matrix, s is scale, t_{2D} is a 2D translation, and Π is an orthographic projection. Our PoseFitter module employs a two-stage process:

Stage 1: Rotation Estimation. We first solve for the rotation matrix R using OpenCV's `cv2.solvePnP` function. This algorithm uses our 3D object points, 2D image points, and an estimated camera intrinsic matrix to return a rotation vector, which is converted to the matrix R.

Stage 2: Scale and Translation Estimation. With R known, we apply it to our 3D landmarks ($P'_{3D} = R \cdot P_{3D}$) and orthographically project them. The problem reduces to finding the optimal scale s and translation t_{2D} that align these projected points with the target 2D landmarks. We solve this efficiently using `cv2.estimateAffinePartial2D`.

This process yields the complete pose parameters, providing a robust initialization for the full model fitting.

2.5. Shape and Expression Fitting

We use a parametric 3DMM (FLAME or BFM), where face shape $S \in \mathbb{R}^{3N}$ is:

$$S = \bar{S} + B_{shape}\alpha + B_{expr}\beta \tag{4}$$

Coefficients:

- $\bar{S} \in \mathbb{R}^{3N}$: mean shape
- $B_{shape} \in \mathbb{R}^{3N \times 300}$: PCA basis for identity-related shape variation
- $\alpha \in \mathbb{R}^{300}$: identity shape coefficients
- $B_{expr} \in \mathbb{R}^{3N \times 100}$: PCA basis for expression deformation

- $\beta \in \mathbb{R}^{100}$: expression coefficients

Landmarks (Dlib 68) are aligned to model vertices via precomputed mappings. The projection follows:

$$p_i = s \cdot \Pi \cdot (R \cdot v_i) + t_{2D} \tag{5}$$

where Π represents the (perspective) camera projection matrix. Camera parameters are estimated using OpenCV's solvePnP, which includes solvers like EPNP, P3P, and DLS.

2.6. Texture Extraction

Vertex colors are obtained via bilinear sampling at the projected vertex positions:

$$c_i = \text{bilinear}(I, \Pi(v_i)) \tag{6}$$

FLAME: always uses image-based sampling; no statistical texture model is used.

BFM: optionally supports a PCA texture model:

$$C = C_\mu + B_{color}\gamma \tag{7}$$

where:

- γ: texture coefficients
- C_μ: mean vertex color (Cmu.npy)
- B_{color}: PCA basis for color

If disabled, BFM also falls back to image-based sampling.

2.7. Energy Minimization

Optimization minimizes the following energy function:

$$E_{total} = E_{landmark} + E_{shape} + E_{expr} \tag{8}$$

where the individual terms are defined as:

$$E_{landmark} = \sum_{i=1}^{K} \|\Pi(v_i) - l_i\|^2 \tag{9}$$

$$E_{shape} = \lambda_s \|\alpha\|^2 \tag{10}$$

$$E_{expr} = \lambda_e \|\beta\|^2 \tag{11}$$

The shape and expression terms use L_2 regularization to keep the face close to the mean. BFM applies separate weights (λ_s, λ_e), while FLAME uses a single weaker weight (λ_{reg}). These weights are scaled by PCA variances (from Ssigma.npy and Esigma.npy) to ensure realistic shapes. Optimization alternates updating α and β, with camera parameters fixed after initialization.

2.8. Rendering and Output

The final mesh is exported as a .obj file, containing:

- Vertex positions S
- Vertex colors C
- Triangle faces (tri.npy or tri_flame.npy)

No lighting or shading is applied. Vertex colors directly represent texture. The output is compatible with visualization tools such as MeshLab. Real-time rendering is not implemented.

2.9. Verification

Once pose parameters are computed, we project the full 3D mean face onto the 2D image and render the wireframe. A close visual match between the wireframe and the subject's facial features indicates successful pose recovery.

(a) Original (b) BFM (c) Flame

Figure 2. Final validation results showing the projected wireframe of the 3D mean face onto the original input image using pose parameters estimated by our pipeline. (a) Original input image, (b) wireframe projection using the BFM, and (c) wireframe projection using the FLAME model. The close alignment between the wireframe and facial features across both models confirms the accuracy of rigid pose estimation.

3. Results and Validation

The primary result of our work is a robustly functioning pose estimation pipeline. The validation is qualitative and visual, as shown in Fig. 2. After the PoseFitter computes the camera parameters, we project the vertices of the entire 3D mean face model onto the image plane and render the wireframe.

As demonstrated in Fig. 2, the resulting wireframe accurately overlays the subject's face. Key features such as the jawline, nose bridge, eyes, and mouth contours show a tight fit. This alignment is the definitive validation for this stage, as an error in any preceding module would result in a visible misalignment. This accurate pose serves as the crucial input for the next stage: optimizing the full energy function. This involves minimizing the reprojection error by adjusting a complete state vector, including not only pose but also shape and expression weights. The optimized wireframe

should then match the person's unique facial structure and expression.

4. Analysis

Our implementation choices reflect a balance between modularity, flexibility, and low-level control. Developing the pipeline in Python enabled direct manipulation of 3D geometry, matrix operations, and numerical optimization routines without relying on external solvers like Ceres. This allowed us to explicitly control projection, regularization, and iterative fitting behavior, making the system easier to debug and extend.

The staged structure of the pipeline—beginning with rigid pose estimation and followed by iterative updates to shape and expression parameters—proved particularly effective. Each step could be visually and numerically validated, which made diagnosing failure cases straightforward. For example, wireframe overlays and landmark reprojection errors were used throughout the optimization loop to track convergence.

The decision to implement custom projection logic and Jacobian computations (rather than rely solely on OpenCV black-box functions) gave us control over numerical stability and allowed us to enforce validity constraints, such as minimum depth and bounding of PCA coefficients. Regularization via L_2 norms further ensured that the recovered identity and expression parameters remained within plausible ranges.

However, the system is not without limitations. Accuracy is strongly dependent on the quality of 2D landmark detection. Inaccurate or occluded landmarks can lead to poor initial alignment, which in turn degrades the optimization. Early stopping criteria is critical for preventing overfitting, as continued optimization beyond convergence often led to unrealistic facial deformations when the solver began fitting to landmark detection noise rather than true geometric structure. Additionally, tuning the PCA bases and regularization weights remains a non-trivial task, especially in the presence of outliers or unusual facial poses.

5. Conclusion

We have presented a clear, modular, and verifiable pipeline for the foundational step of 3D face reconstruction: estimating the head pose from a single 2D image. By decomposing the problem into distinct stages and implementing a visual checkpoint for each, we have built a reliable system that robustly solves for camera rotation, scale, and translation. The successful alignment of the projected 3D mean face demonstrates the efficacy of our approach.

This work provides the essential and validated foundation upon which more advanced reconstruction tasks can be built. Future work will proceed with this robust pose initialization to minimize the full energy function. This will be achieved using a non-linear optimization library like Ceres Solver to find the optimal shape and expression weights that minimize landmark reprojection error, thereby fitting the non-rigid aspects of the face. The final stage will involve texture extraction, where the fitted 3D model is unwrapped to create UV coordinates. By projecting each vertex back into the image plane, we can sample the corresponding pixel color, generating a complete, textured 3D model.

References

[1] Volker Blanz and Thomas Vetter. A morphable model for the synthesis of 3d faces. In *Proceedings of the 26th annual conference on Computer graphics and interactive techniques*, pages 187–194, 1999. 1

[2] Tian Bolkart, Georgios Li, Dimitrios Tzionas, and Michael J. Black. Learning a model of facial shape and expression from 4d scans. *ACM Transactions on Graphics (TOG)*, 36(6):1–17, 2017. 1

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

[4] Pascal Paysan, Rainer Knothe, Brian Amberg, Sami Romdhani, and Thomas Vetter. A 3d face model for pose and illumination invariant face recognition. In *2009 Sixth IEEE International Conference on Advanced Video and Signal Based Surveillance*, pages 296–301, 2009. 1

[5] Justus Thies, Michael Zollhofer, Marc Stamminger, Christian Theobalt, and Matthias Nießner. Face2face: Real-time face capture and reenactment of rgb videos. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2387–2395, 2016.