Computer Vision 2 - Assignment 3
2D to 3D

Hinrik Snær Gudmundsson UvA-id: 12675326

January 7, 2022

1 Introduction

In this assignment, we implement a 3D texture model given a monocular image. We explore how this can be achieved using a combination of a statistical PCA model and energy minimization. We analyze the performance of this method and compare it to previously introduced methods which include ICP and SFM.

2 Morphable Model

Given a prior knowledge of an object, a 3D model can be reconstructed from a single monocular image. One effective approach to capture prior knowledge about an object is to build a statistical model using PCA [2]. For human face modeling, it is possible to represent geometry as a point cloud of N vertices $\boldsymbol{G}(\boldsymbol{\alpha}, \boldsymbol{\delta}) \in \mathbb{R}^{Nx3}$ using a multilinear PCA model [1]:

$$\mathbf{G}(\boldsymbol{\alpha}, \boldsymbol{\delta}) = \boldsymbol{\mu}_{id} + \mathbf{E}_{id}\left[\boldsymbol{\alpha} \cdot \boldsymbol{\sigma}_{id}\right] + \boldsymbol{\mu}_{exp} + \mathbf{E}_{exp}\left[\boldsymbol{\delta} \cdot \boldsymbol{\sigma}_{exp}\right] \tag{1}$$

where $\boldsymbol{\mu}_{id} \in \mathbb{R}^{N\times 3}, \boldsymbol{\mu}_{exp} \in \mathbb{R}^{N\times 3}$ are mean neutral geometry (facial identity) and mean facial expression; $\mathbf{E}_{id} \in \mathbb{R}^{N\times 3\times 30}$ and $\mathbf{E}_{exp} \in \mathbb{R}^{N\times 3\times 20}$ are principal components for neutral geometry and facial expression with their standard deviations $\sigma_{id} \in \mathbb{R}^{30}, \sigma_{exp} \in \mathbb{R}^{20}; \alpha \in \mathbb{R}^{30}$ and $\delta \in \mathbb{R}^{20}$ are latent parameters we need to estimate.

Using the Basel Face Model [3], we implement a morphable 3D face model that uses 30 principal components for facial Identity and 20 principal components for facial expression. If we were to extend the capabilities of our implementation, we would add more principal components which would provide us with the ability to get a wider range of facial identities and/or increase the expressiveness of the generated model.

2.1 Generating 3D Face Model

We uniformly sample $\boldsymbol{\alpha} \sim U(-1, 1)$ and $\boldsymbol{\delta} \sim U(-1, 1)$ to use as input for our implementation.

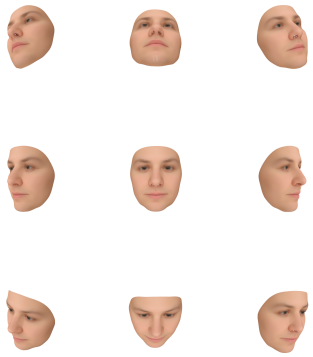


Figure 1: Generated face model when $\boldsymbol{\alpha}$ and $\boldsymbol{\delta}$ are uniformly sampled shown from multiple different angles.

3 Pinhole Camera Model

We have now implemented a function that inputs uniformly sampled $\boldsymbol{\alpha}$ and $\boldsymbol{\delta}$ which provides a 3D face model with facial identity and expression based on the sampled values.

We want to be able to match this 3D face model with a face in a monocular image. We need to convert our 3D model into a 2D projection which can be achieved by using the pinhole camera model. We assume that face vertices $\{x, y, z\} \in \mathbf{G}(\boldsymbol{\alpha}, \boldsymbol{\delta})$ is rigidly transformed using transformation matrix $\Pi \in \mathbb{R}^{4\times4}$ into a camera plane.

$$\begin{bmatrix} \hat{x} \\ \hat{y} \\ \hat{z} \\ \hat{d} \end{bmatrix} = \underbrace{[\mathbf{V}] \times [\mathbf{P}]}_{\Pi} \times \underbrace{\begin{bmatrix} \mathbf{R}(\boldsymbol{\omega}) & \mathbf{t} \\ \mathbf{0} & 1 \end{bmatrix}}_{\mathbf{T}} \times \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \tag{2}$$

Where $\boldsymbol{V}$ is a viewport matrix and $\boldsymbol{P}$ is a perspective projection matrix. Object transformation is modeled using rotation matrix $\mathbf{R}(\boldsymbol{\omega}) \in \mathbb{R}^{3\times3}$ and translation $\boldsymbol{t} \in \mathbb{R}^3$, where $\mathbf{R}(\boldsymbol{\omega})$ is a function over Euler angles $\boldsymbol{\omega} \in \mathbb{R}^3$. Because we want to acquire the U, V projection for our 3D face model instead of coordinates, we can divide $\hat{x}$ and $\hat{y}$ by $\hat{d}$.

The viewpoint transformation matrix encapsulates two stages; 1. a scaling to the shape of the viewport and 2. a transformation to the position of the viewport. We define the viewport as the rectangle between (v_l, v_b) and (v_r, v_t). The scaling matrix is defined by

$$\hat{\mathbf{S}} = \begin{bmatrix} \frac{v_r - v_l}{2} & 0 & 0 & 0 \\ 0 & \frac{v_t - v_b}{2} & 0 & 0 \\ 0 & 0 & \frac{1}{2} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3}$$

And the translation matrix is defined by

$$\hat{\mathbf{T}} = \begin{bmatrix} 1 & 0 & 0 & \frac{v_r + v_l}{2} \\ 0 & 1 & 0 & \frac{v_t + v_b}{2} \\ 0 & 0 & 1 & \frac{1}{2} \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{4}$$

Combining the the two gives us the Viewport Transformation matrix defined by

$$\mathbf{V} = \hat{\mathbf{T}}\hat{\mathbf{S}} = \begin{bmatrix} \frac{v_r - v_l}{2} & 0 & 0 & \frac{v_r + v_l}{2} \\ 0 & \frac{v_t - v_b}{2} & 0 & \frac{v_t + v_b}{2} \\ 0 & 0 & \frac{1}{2} & \frac{1}{2} \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{5}$$

The perspective projection matrix is defined by

$$\boldsymbol{P} = \begin{bmatrix} \frac{2n}{r-l} & 0 & \frac{r+l}{r-l} & 0 \\ 0 & \frac{2n}{t-b} & \frac{t+b}{t-b} & 0 \\ 0 & 0 & -\frac{f+n}{f-n} & -\frac{2fn}{f-n} \\ 0 & 0 & -1 & 0 \end{bmatrix} \tag{6}$$

Where n stands for the near clipping plane and f for the far clipping plane.

3.1 Applying the Pinhole Camera Model

The implementation for the pinhole camera model inputs a 3D face model represented as a collection of {x, y, z, 1} vertices. We also input the viewport (v_l, v_b) and (v_r, v_t), near and far clippling plane f and n, Euler angles $\boldsymbol{\omega}$ and translation $\boldsymbol{t}$.

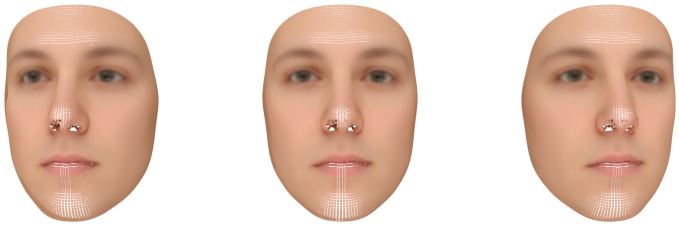


Figure 2: Pinhole Camera Model applied to 3D face model where Euler angles are set to -10, 0 and 10 respectively for the y-axis.

Figure 2 shows the results when using the pinhole camera model where the default parameters are set to $\boldsymbol{t}$ = [0, 0, -500], $\boldsymbol{\omega}$ = [0, 0, 0], f = 0 and n = 0.5. The corresponding z value in $\boldsymbol{t}$ is set to -500 because setting it to 0 will cause the camera to be too close to the model, causing the resulting image to be too wide as can be seen in figure 4. This could possibly be fixed by adjusting the fovy but increasing the distance between the model and the camera is a more elegant solution.

Figure 3: Pinhole Camera Model applied to a 3D face model shown from the side

3.2 Landmark Points

The provided model for 3D face generation also provides the indicies for landmark points that which be extracted. These landmark points can be used for matching landmark points with face keypoints found in a

Figure 4: Pinhole Camera Model applied to a 3D face model where $\boldsymbol{t} = [0, 0, 0]$

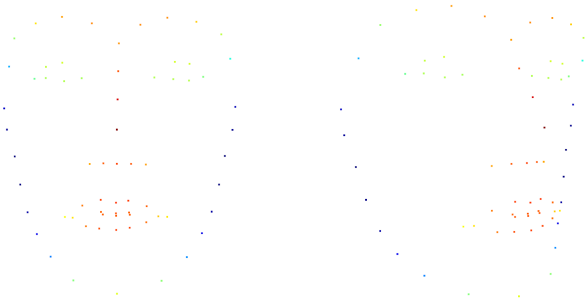


Figure 5: Landmark points extracted from a 3D face model.

monocular image. The provided landmark points are represented by {x, y, z} coordinates but we can apply the pinhole camera model to find the U, V projection for these points.

4 Latent Parameter Estimation

We have built a pipeline which can infer facial landmarks given facial geometry latent parameters $\boldsymbol{\alpha}$, $\boldsymbol{\delta}$ and object transformation $\boldsymbol{\omega}$, $\boldsymbol{t}$. We will estimate these parameters for a monocular image with a human face using Energy minimization. Given 68 ground truth facial landmarks extracted from a monocular image, the following energy can be optimized:

$$\mathcal{L}_{fit} = \mathcal{L}_{lan} + \mathcal{L}_{reg} \tag{7}$$

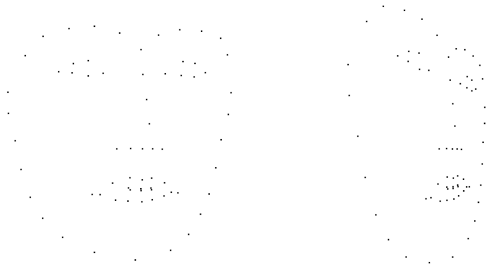


Figure 6: Pinhole Camera applied to landmark points extracted from a 3D face model where Euler angles are set to 10 for the y-axis.

$$\mathcal{L}_{lan} = \frac{1}{68} \sum_{j=1}^{68} \|\mathbf{p}_{kj} - \mathbf{l}_j\|_2^2 \tag{8}$$

where $\mathbf{p}_{k,j} = \left\{u_{k_j}, v_{k_j}\right\}$ is a 2D projection of a landmark point k_j extracted from the 3D face model and $\boldsymbol{l}_j$ is its ground truth 2D coordinate. We regularize the model using Tikhonov regularization to enforce the model to predict faces closer to the mean:

$$\mathcal{L}_{reg} = \lambda_{alpha} \sum_{i=1}^{30} \alpha_i^2 + \lambda_{delta} \sum_{i=1}^{20} \delta_i^2 \tag{9}$$

4.1 Extracting Ground Truth

We want to take a single picture of an individual and apply the provided face landmark detection script on it. This will provide us with the ground truth that is required for the energy optimization.

Figure 7: Face Landmark Detection applied to an image.

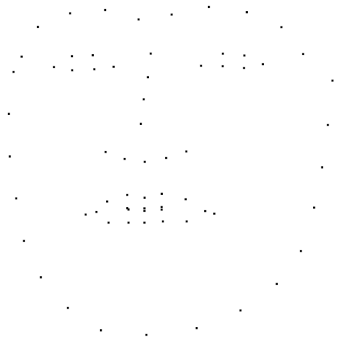


Figure 8: Ground truth extracted from a 2D image

The right image in figure 7 shows that the provided face landmark detection does not perform optimally when facial hair is present. This can also be seen by the size of the chin in figure 8. Nevertheless, we will use this ground truth during training and see how well the energy minimization performs.

4.2 The model

We implement a pytorch module where the input is the far and near clipping plane f and n and the viewport defined by v_l, v_b, v_r and v_t. The input will also include values that will be set as trainable weights, these values are facial expression $\boldsymbol{\delta}$, facial identity $\boldsymbol{\alpha}$, Euler angles $\boldsymbol{\omega}$ and translation $\boldsymbol{t}$.

The forward pass will be based on the pipeline that we previously implemented where we extract the landmark points from a generated 3D face model based on $\boldsymbol{\alpha}$ and $\boldsymbol{\delta}$. We then compute the viewport matrix and the perspective projection matrix based on v_l, v_b, v_r, v_t, $\boldsymbol{\omega}$ and $\boldsymbol{t}$ as described by equations 5 and 6 respectively. We then compute $\{\hat{x}, \hat{y}, \hat{z}, \hat{d}\}$ using equation 2 and then acquire the U, V projection by dividing $\hat{x}$ and $\hat{y}$ by $\hat{d}$.

After we have completed the forward pass, we apply the loss function as described by equation 7 using the 2D projection of the landmark points acquired from the forward pass and the ground truth 2D coordinates acquired from an image. Afterwards the backwards pass is performed and the weights are updated using Adam optimization.

4.3 Hyperparameters

We initialize the model by uniformly sampling $\boldsymbol{\alpha} \sim U(-1, 1)$ and $\boldsymbol{\delta} \sim U(-1, 1)$. Our model has facial hair that effects the ground truth extracted from the image, because of this, we may need to allow the model to be more flexible when it comes to modeling the facial identity. We can allow for this additional flexibility by setting λ_{alpha} to 0.5. We still want the facial expression to be close to neutral so we will enforce a stricter penalty on very expressive facial expression by setting λ_{delta} to 1. Transformation parameters $\boldsymbol{\omega}$ and $\boldsymbol{t}$ will be initialized where the translation over the z dimension is set to -500 and the rest of the values are initialized so that the intitial transformation is as close to the solution as possible i.e. 2D projection of the landmark points after the initial forward pass is as close to the ground truth as possible which will help with convergence. We set the Adam optimizer to train with a learning rate of 0.2 and early stopping at 80.

4.4 Results

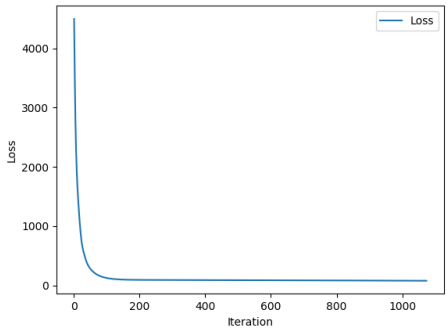


Figure 9: Model loss during training

The model converged after approximately 1100 iterations. As previously observed, the chin of the model increased in size due to the presence of facial hair, this can been seen in figure 10. We adjusted for that by decreasing λ_{alpha} to allow for a wider range of facial identity. We can also observe from figure 10 that an increased λ_{delta} causes the converged model to be more neutral than the facial expression present in the monocular image.

We can now use the learned $\boldsymbol{\alpha}$ and $\boldsymbol{\delta}$ to generate a 3D face model that fits the keypoints from the monocular image best given our energy optimization. The results can be seen in figure 12.

Figure 10: Original image on the left and on the right is the 2D projected 3D model after convergence overlayed on top of the original image

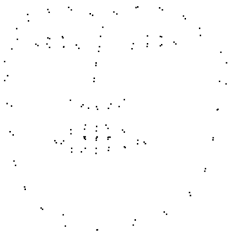


Figure 11: Ground truth and landmark points after convergence

Figure 12: Generated 3D face model after convergence

5 Texturing

We have acquired the latent parameters $\boldsymbol{\alpha}$, $\boldsymbol{\delta}$, $\boldsymbol{\omega}$ and $\boldsymbol{t}$ for the 3D model after performing energy optimization with a monocular image. We can now estimate the color for every individual pixel in our 3D model by performing bilinear interpolation of neighborhood pixels in the original monocular image.

Figure 13: Generated face model using trained $\boldsymbol{\alpha}$ and $\boldsymbol{\delta}$ with colors acquired from bilinear interpolation.

As can be seen from figure 13, no color is present between some points, this leaves white regions that can be seen. We want to texture our model so that color is present despite sparsity of points. This can be achieved by using the triangles that are provided by the 3D face model. We take the uvz vertecies that we have computed along with the estimated colors and pass them into a rendering function along with the provided triangles. The Triangles will be used to interpolate between different points so that colors can be added.

Figure 14: Textured model

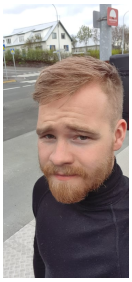
As can be seen on the left side of figure 14, an area beyond the face can be seen. This is consistently with the observation made for figure 7 where the ground truth exceeds slightly beyond the face that is present in the monocular image.

6 Energy optimization using multiple frames

So far we have built a 2D-to-3D reconstruction algorithm which accepts a single monocular image and learns the latent parameters $\boldsymbol{\alpha}$, $\boldsymbol{\delta}$, $\boldsymbol{\omega}$ and $\boldsymbol{t}$ for a 3D model. We want to extend our current model implementation so that we acquire a more accurate facial identity by using M images. This can be achieved by training multiple images in every iteration using a single $\boldsymbol{\alpha}$ along with $\boldsymbol{\delta}_i$, $\boldsymbol{\omega}_i$ and $\boldsymbol{t}_i$ where $i = 1..M$ and indexes the respective image. We then use the index to perform the forward pass using the images deticated weights along with the shared $\boldsymbol{\alpha}$.

We perform the training using the same parameters as we used for the single image implementation with early stopping when the sum of the loss is below 500. the final losses are 123.06, 85.7, 74.9, 148.18 and 68.2 for each image respectively.

(a) Image 1



(b) Image 2

(c) Image 3

(d) Image 4

(e) Image 5

Figure 15: Images that will be used during training

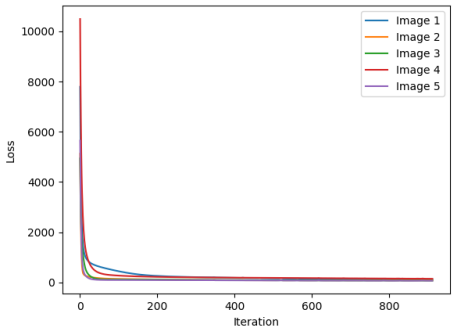


Figure 16: Model loss during training

The results found in figure 17 show that energy optimization using multiple frames achieves a more accurate facial identity, this can be seen when comparing the results for image 3 when it was trained alone and when it was trained along with multiple frames. The results still however vary which can be attributed to difficult angles and the presence of facial hair which can make it more difficult to extract accurate ground truths. Setting the early stopping threshold at a lower value is likely to produce better reconstructions.

(a) Image 1

(b) Image 2

(c) Image 3

(d) Image 4

(e) Image 5

Figure 17: Textured model after convergence

7 Discussion

Throughout this course, we have explored various methods to implement a depth based 3D reconstruction.

ICP proves to be useful when multiple pointclouds have been gathered from various angles and the goal is to make a single 3D reconstruction. ICP iteratively merges the different angles by finding the best rotation and translation based on mutual keypoints. ICP is however susceptible to statistical outliers and trade-offs are to be made when selecting efficient variants of the ICP algorithm. Statistical outliers can have a significantly large impact on the results of the algorithm, causing results to be sub optimal when the algorithm is applied on a pointclouds that contain statistical outliers.

SFM proves to be a useful tool when 3D modeling an object that is present throughout multiple frames. Where ICP is well suited when using multiple individual and possibly independent frames of an object, SFM is better suited when using multiple consecutive frames from the same source e.g. 3D reconstrucition from a video recording. SFM can be very effective but limited overall in its application compared to its ICP counterpart. SFM requires more certain criteria to be met to be effective, such as consecutive frames from the same source where motion is present while also having sufficient mutual keypoints throughout every consecutive frame.

ICP and SFM can be combined to acquire a 3D reconstruction when the criterias for both ICP and SFM are met. One example is when there are N recordings of an object from different sources. SFM can be applied individually to every recording, this will provide us with N pointclouds. ICP can then be applied to combine these N pointclouds into a single 3D reconstruction of the object.

The biggest advantages of the 3D texture model is that we are able to acquire a 3D reconstruction based on a single monocular image, this is something that cannot be achieved with a single image using ICP or SFM. Another big advantage over ICP and SFM is that the PCA model provides triangles that can be used for texturing, allowing us to acquire a fully textured model instead of being limited by the density of the pointcloud. One limitation however is that the 3D reconstruction is limited by how well the PCA model can be fitted on a monocular image. If the PCA model is not able to accurately match the face in the monocular image, either due to extreme facial identity or facial expression, then we will not be able to acquire a satisfactory 3D reconstruction. The 3D reconstruction is also limited by what the PCA model is able to generate i.e. we cannot perform a 3D reconstruction of objects or body parts that exceed the capabilites of the PCA model. This is a limitation that is not present in ICP and SFM.

ICP and/or SFM can be used with our 3D texture model where the 3D texture model exclusively reconstructs the face. ICP and/or SFM is used to reconstruct everything that exceeds the capabilities of the PCA mode e.g. the rest of the body. We can apply ICP and/or SFM as previously described but now we can also include the pointcloud that we acquire using the 3D texture model. This would allow us to perform full body reconstruction with a detailed face.

References

[1] V. Blanz and T. Vetter. A morphable model for the synthesis of 3d faces. In *Proceedings of the 26th Annual Confrence on Computer Graphics and Internative Techiques*, 1999.

[2] T.F. Cootes. An introduction to active shape models. 1992.

[3] T. Gerig, A. Morel-Forster, C. Blumer, B. Egger, M. Luthi, S. Schoenborn, and T. Vetter. Morphable face models - an open framework. In *2018 13th IEEE International Conference on Automatic Face Gesture Recognition (FG 2018)*, pages 75–82, 2018.