3D Morphable Model

Submitted By: Gurkirat Singh Bajwa
Sahil Aggarwal

Introduction

The 3D morphable model was introduced by Volker Blanz and Thomas Vetter in [BV99] in 1999. The paper demonstrates a technique which can be used to:

- Generate human-like faces easily.
- Interpolate between two arbitrary faces.
- Change facial attributes like age, gender, weight etc.
- Accurately modelling 3D faces from single 2D images.

Database

We used the database provided by the original authors.

- The data set provides was created using Laser Scans (*Cyberware*^{TM}) of 200 heads of young adults(100 male and 100 female).
- Additionally, RGB values were recorded in same spatial information and were stored in a texture map with 8 bits per channel.
- Data contains 53,490 vertices and 106,466 triangles per face.
- 4 manually labelled attributes, namely; Age, Gender, Height and Weight.

The Model

Each face is represented by a tuple of shape and texture vectors ***(S,T)***.

Where, $S = (X_1, Y_1, Z_1, X_2, Y_2, \ldots\ldots\ldots\ldots, X_n, Y_n, Z_n) \in R^{3n}$

And $T = (R_1, G_1, B_1, R_2, G_2, \ldots\ldots\ldots\ldots, R_n, G_n, B_n) \in R^{3n}$

Since we have m (= 200) faces in full correspondence, we can represent any arbitrary face as a convex combination of all the faces we have:

$$S_{mod} = \sum_{i=1}^{m} a_i S_i \; ; \; T_{mod} = \sum_{i=1}^{m} b_i T_i \; ; \; \sum_{i=1}^{m} a_i = 1; \; \sum_{i=1}^{m} b_i = 1$$

The Model Improved

We estimate the distribution of variables a_i and b_i in our model.

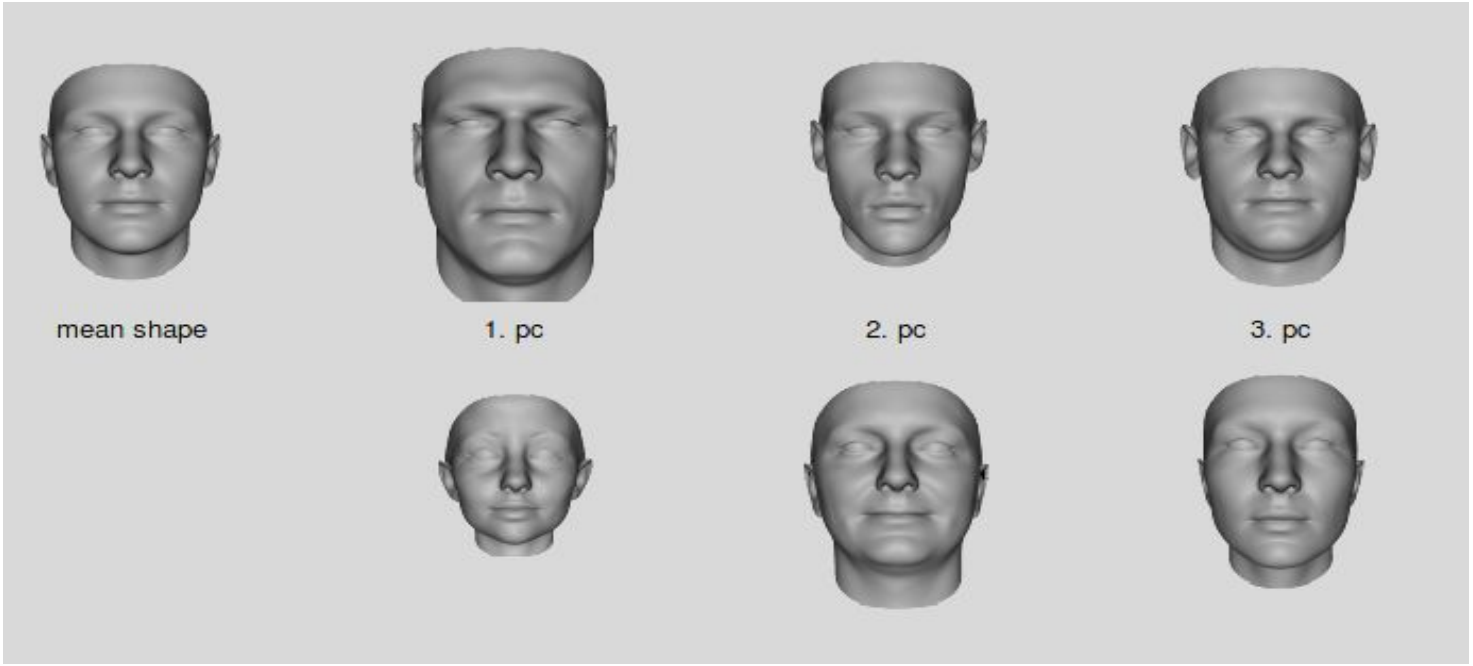
We fit a multivariate gaussian distribution to our data of 200 faces based on average shape $\overline{S}$ and average texture $\overline{T}$ and covariance matrix C_S and C_T computed over shape and texture difference $\Delta S_i = S_i - \overline{S}$ and $\Delta T_i = T_i - \overline{T}$.

We do a PCA to do a basis transformation to an orthogonal system of eigen vectors s_i and t_i of covariance matrices.

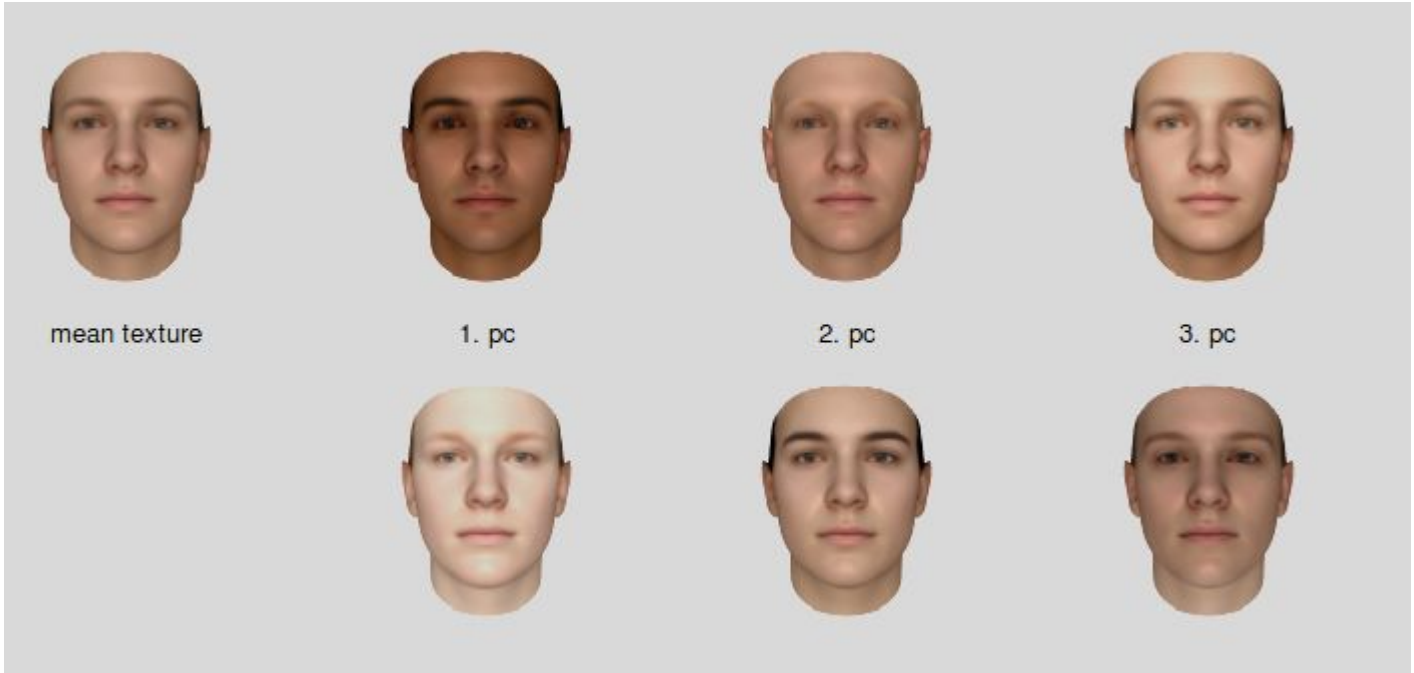
$$S_{mod} = \overline{S} + \sum_{i=1}^{m-1} \alpha_i S_i \; ; \; T_{mod} = \overline{T} + \sum_{i=1}^{m-1} \beta_i T_i \; ; \; \alpha, \beta \in R^{m-1}$$

$$p(\alpha) = exp[\frac{-1}{2} \sum_{i=1}^{m-1} \frac{\alpha_i^2}{\sigma_i^2}]$$

The Mean and first 5 principal components of shape for model.



The Mean and first 5 principal components of shape for model.



Facial Attributes

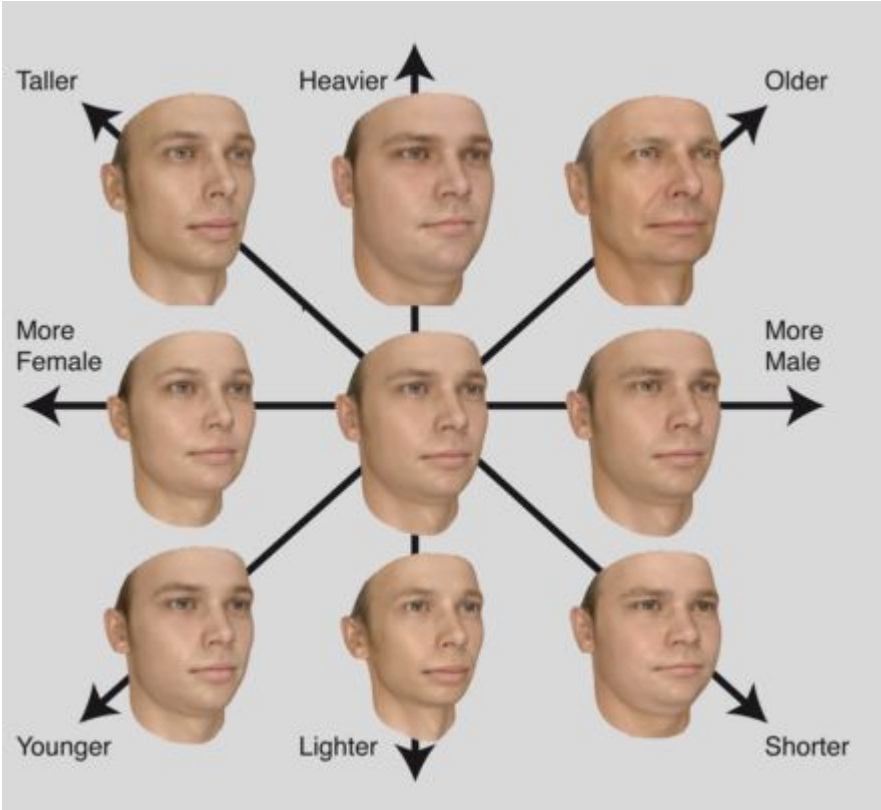
Based on a set of faces (S_i,T_i) with manually assigned labels μ_i describing the markedness of the attributes, we compute the weighted sums

$$\Delta S = \sum_{i=1}^{m} \mu_i (S_i - \overline{S}); \quad \Delta T = \sum_{i=1}^{m} \mu_i (T_i - \overline{T})$$

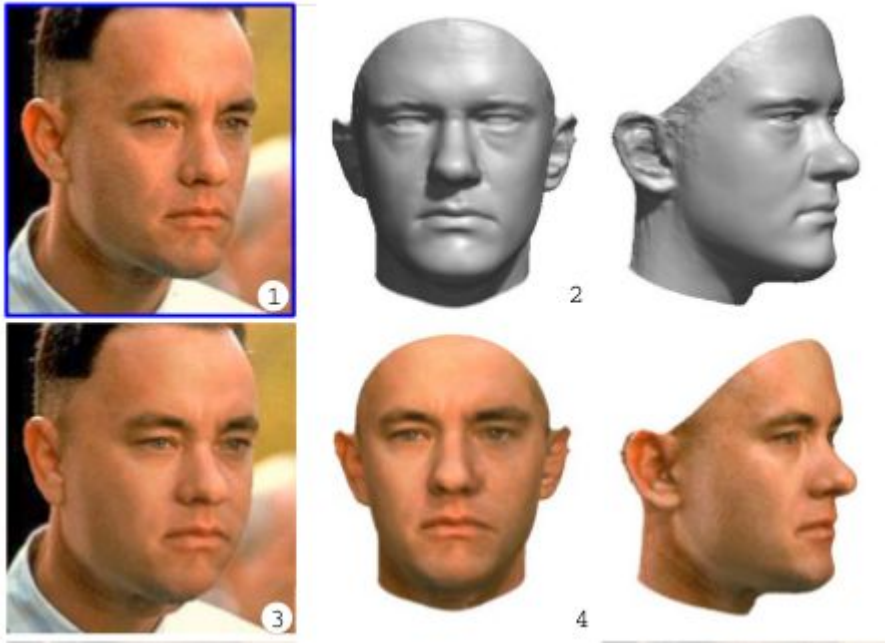
Now multiples of $(\Delta S,\Delta T)$ can be added or subtracted from any individual face.

For facial expression, we can similarly compute

$$\Delta S = S_{expression} - S_{neutral} \quad \Delta T = T_{expression} - T_{neutral}$$


Taller
Heavier
Older
More
Female
More
Male
Younger
Lighter
Shorter

Fitting Model to an Image



Fitting Model to images

Rough initial manual alignment of model onto target image.

Coefficients of the 3D model $\overline{\alpha}$, $\overline{\beta}$ are to be optimized with rendering parameters $\overline{\rho}$ such as camera position, object scale, image plane rotation, translation, intensity of ambient light and directed light etc.

$$\overline{\alpha} = (\alpha_1, \alpha_2, \ldots\ldots\ldots\ldots\ldots\ldots \alpha_{199})$$

$$\overline{\beta} = (\beta_1, \beta_2, \ldots\ldots\ldots\ldots\ldots\ldots \beta_{199})$$

Fitting Model to images

At every iteration, the algorithm renders an image I_{model} using the current model parameters $\overline{\alpha}$, $\overline{\beta}$ and rendering parameters $\overline{\rho}$ and updates them to minimize the norm of residual image.

$$E_I = \sum_{x,y} ||I_{input}(x,y) - I_{model}(x,y)||^2$$

Rendering is performed using perspective projection and phong illumination model. Projection maps the 3D model to locations *(x,y)* in above model and shading algorithm computes the color values at those pixels.

Fitting Model to images

- To force the solution set to lie within the set of faces(close to the mean values) we maximise the posterior probability $p(\overline{\alpha}, \overline{\beta}, \overline{\rho})$ given I_{input}. The distributions are assumed to be gaussian.
- Also, the noise in I_{input} is assumed to be gaussian with mean zero and standard deviation σ_N. So the distribution of the difference between rendered model and input image is modelled as

$$exp(-E_i/2\sigma_n^2)$$

Fitting Model to images

Maximising posterior probability is then equivalent to minimising the cost function given by

$$E = \frac{E_I}{2\sigma_N^2} + \sum_{j=1}^{m-1} \frac{\alpha_j^2}{\sigma_{S,j}^2} + \sum_{j=1}^{m-1} \frac{\beta_j^2}{\sigma_{T,j}^2} + \sum_j \frac{(\rho_j - \overline{\rho}_j)^2}{\sigma_{\rho,j}^2}$$

Here, $\sigma_{S,j}$ and $\sigma_{T,j}$ is the standard deviation for shape and texture obtained from PCA previously. $\sigma_{\rho,j}$ is the standard deviation for rendering parameters obtained by ad-hoc.

The cost function is equivalent to minimising log likelihood of posterior probability.

Fitting Model to images

We optimise the cost function using Batch Gradient Descent. The partial derivatives can be found analytically and the partial $\partial E_I/\partial \alpha_i$ can be estimated using finite differences.

$$\alpha_i = \alpha_i - \lambda_i \frac{\partial E}{\partial \alpha_i}$$

A coarse to fine strategy is used:

- First iterations are performed on a low resolution model and subsampled I_{input}.
- The highest principal components are used first and rest are used later.

Key Takeaways

- Learned 3D rendering in Matlab and PLY format for 3D representation.
- Initialisation holds key in an optimisation problem.
- Convergence is a difficult thing to achieve.
- Not every implementation available on github is valid.

Todo

- Segment wise-fitting to solve optimisation problem.
- Some more experimentation with learning rates and a better estimate of rendering parameters.

Thank You