Active Appearance Models for Face Modelling

CS 269 Course Project

Shubham Mittal
UID: 104774903
University of California, Los Angeles
mitshubh@cs.ucla.edu

Taylor Caulfield
UID: 404773737
University of California, Los Angeles
taylorcaulfield@g.ucla.edu

***Abstract* — Face modelling has long been a target of research in the realm of computer science due to its many practical applications. However, accurately modelling a face in an image can be both difficult and computationally expensive. One potential avenue for effectively creating a face model from an image is the application of Active Appearance Models (AAM) - a generative, deformable, statistical-based template matching method. The base face model is generated from a series of input data points. To apply this model to an image, we split an image into its shape and texture data, perform Generalized Procrustes Analysis on the shape data, and perform a piece-wise affine warp on the texture data. We test this method on the challenging Helen image dataset (hosted by iBug) and show how Active Appearance Models can efficiently fit images using a pre-trained model.**

Keywords — Active Appearance Models, Delaunay Triangulation, Generalized Procrustes Analysis, Principle Components Analysis

I. Contents

II. Introduction

The creation of accurate models of faces in photographs and video has been the subject of heavy research due to their wide range of applicability in other fields such as psychology and social sciences. A face model can capture the emotions of an image subject, which can be used to assess the communicative intent of a political image or evaluate the satisfaction of a consumer with a particular program, among other things. In addition, face models can be altered to create new faces, which can be used to generate new images or video.

Unfortunately, there are many hurdles in creating an accurate face model, which often relies on identifying facial features such as eyes, ears, and mouths. Many of these challenges occur in the facial feature identification phase. The potential variations of the appearance of a face in an image can make it difficult to draw out a general rule of thumb for identifying a particular feature on a face. For one, the structure and tone of a person's face greatly varies from person to person, resulting in a wide range of possible shapes for a given facial feature. The pose and orientation of a face can also skew the dimensions and positioning of the features of the captured face. Additionally, the medium used to capture face can impose its own alterations on the image, causing variations in brightness, contrast, sharpness, color balance, and quality. The computation of a face model can also be time-consuming and expensive.

One promising solution to overcome these hurdles is to apply an Active Appearance Model (AAM) to a facial image. An Active Appearance Model is a generative, non-linear, deformable model that represents a particular visual structure (e.g. a face, a house, etc.) [3][4]. This process consists of two main steps; building the model from a set of training images, and fitting the model to an incoming image. Once fitted to an image, the model's parameters can be extracted and can be analyzed for data or manipulated to produce new models or media

III. Methodology

A. Modelling

An Active Appearance Model consists of two models: the shape model and the texture model. The shape model is created by capturing the shape data from a set of training faces, aligning the data into a common frame, and analyzing the variance within the newly-aligned data.

To capture a face's shape data, the landmarks on a face must be identified. Landmarks are points on the image that delineate the boundaries of facial features. Each landmark is given a numeric label, and its coordinates are bundled into a vector of size n*k, where n is the number of landmarks and k is the number of dimensions in the image. For a 2D image with n landmarks, this vector is defined as follows:

$$s = (x_i y_i | 1 < i < n)^T = (x_1 y_1 x_2 y_2 x_3 y_3 \ldots\ldots\ldots x_n y_n)^T \quad (1)$$

Sets of shapes can be aligned into a common frame via General Procrustes Analysis, which utilizes an alignment technique known as Procrustes Analysis to sequentially align each shape in the set to a running average.

The Procrustes Analysis technique aligns two shapes with the same number of landmarks and one-to-one point correspondence. It consists of the steps listed below:

1. Compute the shape centroids
2. Align the centroids of both shapes to the origin
3. Normalize both shapes
4. Arrange the shape matrices
5. Use re-arranged shape matrices to perform SVD
6. Obtain Rotation Matrix from SVD

The shape centroid of a shape s is a point whose x and y coordinates are the mean of all the x and y coordinates of the shape's n landmarks, respectively:

$$\bar{s} = (\overline{s_x}, \overline{s_y}) = (\frac{1}{n}\sum_{i=1}^{n} x_i, \frac{1}{n}\sum_{i=1}^{n} y_i) \quad (2)$$

Aligning the centroid of the shape is performed as below:

$$s_c = s - \bar{s} \quad (3)$$

Normalizing the shape can be done with isomorphic scaling, as shown below:

$$\hat{s} = \frac{s_c}{||s_c||} \quad (4)$$

For each normalized shape, we transpose the x and y coordinates into x and y columns respectively and concatenate the y column to the right of the x column to obtain the shape matrix S, a n x 2 matrix:

$$S = [\hat{s}_x | \hat{s}_y] \quad (5)$$

Given two shapes with shape matrices S_1 and S_2, we perform SVD on the two matrices:

$$SVD(S_1 S_2 = U \Sigma V^T) \quad (6)$$

From the result, we can obtain the rotation matrix R, which represents the best alignment of S_2 to S_1. [2]

$$R = UV^T \quad (7)$$

General Procrustes Analysis applies Procrustes Analysis to each shape in the target set using a reference shape, a "running average" of all shapes in the set that have been processed.

- Choose first shape as estimate for mean shape ($\bar{x}_0$)
- k = 0
- repeat until mean converges ($\bar{x}_{k+1} \approx \bar{x}_k$)
    - For each of the m shapes x, Procrustes($x, \bar{x}_k$)
    - k+=1
    - Recompute mean ($\bar{x}_k$) with re-aligned images

Once the shape sets are aligned with the method above, we run Principle Components Analysis (PCA) on the shapes. Given m shape sets s with n landmarks each, the shape sets will be represented in a space with a number of dimensions d equal to the size of the shape sets. For a 2D image, each shape set has 2n points, so d = 2n. Note that m must be larger than the size of the shape set, so for a 2D image, m > 2n.

For each of the d dimensions, PCA calculates an eigenvector that points in the direction in the data space with the largest variance amongst the data points, along with its associated eigenvalue. The eigenvalues are ordered by decreasing magnitude (amount of variance).

PCA is performed as follows:

- Compute the shape average:

$$\bar{s} = \frac{1}{m}\sum_{i=1}^{m} s_i \quad (8)$$

- Compute maximum likelihood estimation of covariance matrix:

$$C = \frac{1}{m-1}\sum_{i=1}^{m}(s_i - \bar{s})(s_i - \bar{s})^T \quad (9)$$

- Compute eigenvectors and eigenvalues from covariance matrix
- Create orthogonal matrix $\boldsymbol{\Phi}$ from t most important eigenvectors: $\Phi = (\phi_1 | \phi_2 | \ldots | \phi_t)$
    - t is the number of modes of variation is chosen such that $\sum_{i=1}^{t} \lambda_i \geq p \sum_{i=1}^{m} \lambda_m$
        - p is a constant signifying some portion of the sum of all eigenvalues; usually, $p \geq .90$
- Compute t-dimensional projection vector for a given shape s:

$$b = \Phi^T(s - \bar{s}) \quad (10)$$

From this projection vector, we can compute the statistical shape variation for a given shape s:

$$s = \bar{s} + \Phi_s b_s \quad (11)$$

Texture information is represented by the intensity of each pixel in the face. Pixel intensities are captured in a column vector c, which contains the intensities for each of the three color channels - red, green, and blue - of a given pixel. Given m pixels sampled, the vector v will be of size 3m and will be

as follows:

$$v = (\{ r_i, g_i, b_i \mid 1 < i < n\})^T = (r_1, g_1, b_1, ..., r_n, g_n, b_n)^T \quad (12)$$

To build a texture model, a warping function for each training image is computed. This function moves a set of corresponding control points to match points in a mean shape, which is represented as a Delaunay Triangulation. Usually, a reverse mapping, which transforms the target points into the original points, is performed alongside bilinear interpolation correction. This procedure is carried out with a Piece-wise Affine Warp, which proceeds as follows:

- Partition the convex hull of the mean shape into a set of triangles using Delaunay Triangulation
- for each control point x in the mean shape:
  - find x's corresponding triangle t
  - get the relative position of x in its triangle
  - use bilinear interpolation correction to map x

The corresponding triangle for a point can be determined using its barycentric coordinates. Given a control point x and a triangle t with vertices x1, x2, and x3, the barycentric coordinates α, β, and γ can be obtained from the following equations:

$$\begin{aligned} \alpha &= 1 - (\beta + \gamma) \\ \beta &= \frac{yx3 - x1y - x3y1 - y3x + x1y3 + xy1}{-x2y3 + x2y1 + x1y3 + x3y2 - x3y1 - x1y2} \\ \gamma &= \frac{xy2 - xy1 - x1y2 - x2y + x2y1 + x1y}{-x2y3 + x2y1 + x1y3 + x3y2 - x3y1 - x1y2} \end{aligned} \quad (13)$$

If $0 \leq \alpha, \beta, \gamma \leq 1$, then x falls inside t.

Finally, two linear interpolations on x are performed using its four neighbours to obtain the interpolation value F. Given a control point with coordinates x_x and x_y and neighbors F_1, F_2, F_3, and F_4, F is calculated as follows:

$$F = [1 - x_x \;\; x_x][F_1 \;\; F_2][1 - y] \quad (14)$$
$$[F_3 \;\; F_4][\;\; y \;\;]$$

A statistical texture model can be obtained by performing PCA on the intensity vector v using the mean texture vector and the orthogonal matrix of the t most important texture eigenvectors $\boldsymbol{\Phi}_v$. The vector of texture parameters b_v is calculated as follows:

$$b_v = \Phi_v{}^T (v - \bar{v}) \quad (15)$$

To combine the shape and texture models, a third PCA is performed to remove any correlations between the parameters of the two models. This results in the combined model shown below:

$$b = \begin{pmatrix} W_s b_s \\ b_g \end{pmatrix} \quad (16)$$

W_s is a diagonal matrix that carries the differences between the texture and shape parameters.

This combined model can be used to extrapolate any shape or texture value from the mean shape or texture, respectively:

$$s = \bar{s} + \Phi_s W_s^{-1} \Phi_{cs} c \quad (17)$$
$$v = \bar{v} + \Phi_v \Phi_{cv} c \quad (18)$$

where Φ_{cs} and Φ_{cv} hold the tc most important eigenvalues for the shapes and textures, respectively, and

$$c = \begin{pmatrix} (\Phi_{cs})^T b \\ (\Phi_{cv}) \end{pmatrix} \quad (19)$$

B. Face Fitting

Before our combined model can be modelled to an image with a face, the face's location in the image must first be pinpointed, which we perform using AdaBoost [5]. AAM model fitting is a non-linear optimization problem. Our approach to optimizing our AAM model is a gradient descent approach. Like the similar Lucas-Kanade approach, we recover the parametric description of a face instance. However, registration from the input image is obtained through a parametric appearance model rather than a static template [1].

IV. EXPERIMENTATION

We tested a C++ implementation of our AAM modelling and fitting procedures coupled with OpenCV using subsections of the challenging Helen image dataset hosted by iBug. Our training set consisted of 2000 images that came in differing resolutions and image quality, showcasing a wide variety of face types and poses. This ensured that any AAM model generated would not be too specific to a set of similar-looking images. Each image in the Helen image set is accompanied by an annotation file with 68 point annotations denoting landmarks in the image. A snippet of an annotation file is shown in Fig1:

```
version: 1
n_points:  68
{
337.686269 431.562644
326.604425 505.028314
325.378133 577.803269
330.459111 650.023914
…
}
```

Fig. 1. Annotation File consisting of 68 points

To detect facial positions in our images, we utilized a Tree-based 20x20 Gentle Adaboost Frontal Face Detector [6].

Training the model using larger images resulted in unreasonably-long training times, so we stuck with smaller images for the training process. Even so, the training process took 20 minutes on average for a set of 100 images

(a)

(b)

Fig. 2. (a) Sample training images in Helen dataset, (b) Combined AAM instances generated

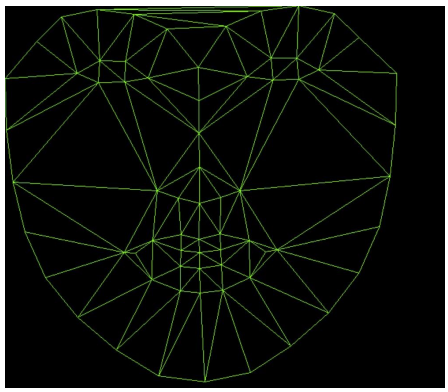

Fig. 3. Delaunay triangulation of a generated AAM instance, totaling at 113 triangles

Fig 2 shows the AAM instances extracted from the trained images. As can be seen, the image dataset used is very challenging, but we are able to extract good AAM instances using only the points data. The fitting process was done on an image by image basis. This process was generally very quick; it often took just 200-300 milliseconds to fit any image, regardless of size. Fig. 4 shows the experimental results of our implementation of the AAM model. Fig. 4 shows the cases where our implementation did not perform well. There were some cases where our implementation did not perform well. While the mean shape could rotate and translate well, it would often have a number of textures deformations due to the high variance of texture maps from the incoming testing images (e.g. from spectacles and from objects blocking out parts of the face).

(a)

(b)

Fig. 4. (a) Sample testing images in the Helen dataset, (b) Fitting results on the testing images

Conclusions

Modelling faces effectively and accurately can be difficult, but we believe our solution tackles the problem fairly well. Using a set of 2000 training images, our method trains two different models. One is a shape model calculated from the results of Generalized Procrustes Analysis performed over the shape data of the sample data. The other is a texture model created from a mapping of the texture information created from the Piece-wise Affine Warp of the images' Delaunay Triangulations. These models are then combined to create an AAM. While the training procedure was slow, it yielded decent AAM models. Fitting our models to our set of testing images was quick and yielded acceptable (albeit noticeably warped) results.

References

[1] Antonakos, Epameinondas, et al. "Feature-based lucas–kanade and active appearance models." IEEE Transactions on Image Processing 24.9 (2015): 2617-2632. APA.

[2] Bookstein, Fred L. "Landmark methods for forms without landmarks: localizing group differences in outline shape." Proceedings of the IEEE workshop on mathematical methods in biomedical image analysis. Vol. 96. IEEE Computer Society, 1996.

[3] G.J. Edwards T.F.Cootes and C.J.Taylor. Active appearance models. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2001.

[4] Matthews, Iain, and Simon Baker. "Active appearance models revisited." International Journal of Computer Vision 60.2 (2004): 135-164.

[5] Viola, Paul, and Michael Jones. "Rapid object detection using a boosted cascade of simple features." Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on. Vol. 1. IEEE, 2001.

[6] Elmer, Peter, et al. "Exploring compression impact on face detection using haar-like features." Scandinavian Conference on Image Analysis. Springer International Publishing, 2015