An Experiment on 3D Face Model Adaptation using the Active Appearance Algorithm

Jörgen Ahlberg
ahlberg@isy.liu.se
Image Coding Group, Dept. of Electrical Engineering,
Linköping University, SE-581 83 Linköping, SWEDEN
http://www.icg.isy.liu.se

Abstract

This report describes our first implementation and experiment on adapting a 3D wireframe model to a face in an input image using the Active Appearance Algorithm. The results are promising, and it is our conclusion that we should continue on this track in our task to create a real-time model-based coder.

1. Introduction

Our goal is to find and track a human face and its features in a video sequence. We want to do this by adapting, in each frame, a wireframe model to the face in the image. This should be done accurately enough to allow a realistic-looking facial animation to be created from the extracted face model parameters, and it should also have real-time performance. We use the CANDIDE-3 [4] model since it is simple, but yet able to represent all the parameters we wish to extract. This report describes our first try to use the Active Appearance Algorithm to adapt the model to the image.

The concept of Active Appearance Models (AAMs) was introduced a few years ago [2], and has been the subject of several reports and investigations, especially by the original inventors [1]. Together with the AAMs came a directed search algorithm for adapting the model to an image, here referred to as the Active Appearance Algorithm (AAA). The AAA can be used on a complete AAM, or, as here, on a simpler model being parameterized in geometry and texture.
The AAA is a greedy algorithm, finding locally optimal face model parameters. Thus, a quite accurate à priori estimate of the face model parameters (at least the size and position of the face) must be available before starting the AAA. In a video sequence, we can use the parameters extracted from the previous frame as the à priori estimate. For the first frame of the sequence, we need an additional algorithm, for example a colour or motion based face candidate finder. Such algorithms are not treated in this report.

This report describes our first implementation and experimentation with the AAA. During the implementation process, there has been a lot of design decisions not mentioned here, with the overall goal to as fast as possible have a running program. Therefore, there are several things that would look (and later will look) different if time had allowed.

Section 2 treats the parameterization of the face model and how to find the optimum parameters using the AAA. In the following two sections, the training and testing processes as performed in this experiment are described. In Section 5, a few directions of our continuing work are mentioned.

2. Model Parameterization and the Active Appearance Algorithm

The model is a wireframe model with a texture mapped on its surfaces. The texture is represented as a standard-shaped image, being a linear combination of a set of Texture Units (TUs) or *eigenfaces* [3]. We formulate this as

\[ x = \bar{x} + X\tau \quad (1) \]

where \( \bar{x} \) is the mean texture, the columns of \( X \) are the TUs and \( \tau \) is the vector of texture parameters. The synthesized texture \( x \) is mapped on the wireframe model, which is then reshaped according to

\[ g(\sigma, \alpha) = \bar{g} + S\sigma + A\alpha \quad (2) \]

where the resulting vector \( g \) contains the \((x, y, z)\) coordinates of the vertices of the model. \( \bar{g} \) is the standard shape of the model, and the columns of \( S \) and \( A \) are the Shape and Animation Units respectively, and thus \( \sigma \) and \( \alpha \) contain the shape and animation parameters.

Since we also want to perform global motion, we need six more parameters for rotation, scaling, and translation. Thus, we replace (2) with

\[ g = sR(\bar{g} + S\sigma + A\alpha) + t \quad (3) \]

or

\[ g = sR(\bar{g} + G\gamma) + t \quad (4) \]
where \( \mathbf{R} = R(r_x, r_y, r_z) \) is a rotation matrix, \( s \) is the scale, and \( \mathbf{t} = t(t_x, t_y) \) the translation vector, \( \gamma \) is the concatenation of \( \sigma \) and \( \alpha \) and \( \mathbf{G} = [\mathbf{S} \, \mathbf{A}] \).

The geometry of our model is thus parameterized by the parameter vector
\[
\mathbf{p} = \begin{bmatrix} \mathbf{v}^T \; \mathbf{y}^T \end{bmatrix} = \begin{bmatrix} \mathbf{v}^T \; \sigma^T \; \alpha^T \end{bmatrix}
\]
where \( \mathbf{v} \) is the vector of global motion parameters.

Note that this differs from the original AAM formulation, where out-of-plane rotation is rather built-in in the Shape Units than a separate parameter. Another difference is that in the original formulation, there is no distinction between Shape Units and Animation Units.

When adapting a model to a video sequence, the shape parameters \( \sigma \) should only be changed in the first frame(s); the head shape does not vary during a conversation. The shape parameters can be converted to MPEG4 Facial Definition Parameters (FDPs) and the animation parameters to MPEG4 Facial Animation Parameters (FAPs).

### 2.1. Matching the model and the image

Our goal is to find the optimal adaptation of the model to the input image, that is to find the \( \mathbf{p} \) that minimizes the distance between the model and the image. As distance measure, we choose the summed squared error (SSE) between the remapped input image and the synthesized texture. We compute this by, for a given \( \mathbf{p} \), reshaping the model according to \( \mathbf{g}(\mathbf{p}) \), and mapping the input image \( \mathbf{i} \) onto the model. We then reshape the model to the standard shape, \( \bar{\mathbf{g}} \), and get the resulting image as a vector \( \mathbf{j} = \mathbf{j}(\mathbf{i}, \mathbf{g}(\mathbf{p})) \). This image can be approximated by the Texture Units according to (1), and we compute the residual image
\[
\mathbf{r} = \mathbf{j} - \mathbf{x},
\]
and thus the SSE as
\[
e = \|\mathbf{r}\|^2
\]

The optimal \( \mathbf{x} \) (minimizing \( e \)) is given by the parameter vector \( \mathbf{t} \) computed as
\[
\mathbf{t}(\mathbf{j}) = \mathbf{X}^T (\mathbf{j} - \mathbf{x}),
\]
so that,
\[
\mathbf{x}(\mathbf{j}) = \bar{\mathbf{x}} + \mathbf{XX}^T (\mathbf{j} - \bar{\mathbf{x}}),
\]
and we can consequently write the residual image and the SSE as functions \( \mathbf{r}(\mathbf{p}) \) and \( e(\mathbf{p}) \) of the model parameters \( \mathbf{p} \).
2.2. Finding the optimal parameters

With this formulation, our goal is to find the parameter vector $p$ that for a given input image $i$ minimizes $e(p)$. We do that by using the Active Appearance Algorithm (AAA) in the following way. For a starting value of $p$, supposed to be close optimum, we compute $r(p)$ and $e(p)$, and find the update vector $\Delta p$ by multiplying the residual image with an update matrix:

$$\Delta p = U r$$  \hspace{1cm} (10)

The vector $\Delta p$ gives us a probable direction in the search space, and we the compute the SSE

$$e_k = e(p + k\Delta p)$$  \hspace{1cm} (11)

for a few values of $k$. We let

$$k^* = \arg\min_k e_k. \hspace{1cm} (12)$$

and update $p$ accordingly, that is,

$$p + k^*\Delta p \rightarrow p. \hspace{1cm} (13)$$

and iterate until convergence. The magic in this is the update matrix $U$, that we create in advance by training from example images with models correctly adapted.

2.3. Creating the update matrix

Assuming that $r$ is linear in $p$, that is,

$$\frac{\partial}{\partial p} r(p) = R$$  \hspace{1cm} (14)

where $R$ is constant, we can write

$$r(p + \Delta p) = r(p) + R\Delta p. \hspace{1cm} (15)$$

Given a $p$ (and thus an $r(p)$), we want to find the $\Delta p$ that minimizes

$$e(p + \Delta p) = \|r(p) + R\Delta p\|^2. \hspace{1cm} (16)$$

The least square solution is

$$\Delta p = -(R^T R)^{-1} Rr(p)$$  \hspace{1cm} (17)

which gives us the update matrix $U$ as the negative pseudo-inverse of the gradient matrix $R$:

$$U = -R^+ = -(R^T R)^{-1} R$$  \hspace{1cm} (18)
To be able to use the AAA, we should consequently estimate the gradient matrix $\mathbf{R}$. We do that by perturbing $\mathbf{p}$ from a set of (manually) adapted models parameter by parameter, step by step. The $j$:th row in $\mathbf{R}$ can thus be estimated as

$$
\mathbf{R}_j = \sum_k (r(\mathbf{p} + \Delta \mathbf{p}_{jk}) - r(\mathbf{p}))
$$

(19)

where $\Delta \mathbf{p}_{jk}$ is a vector that perturbs $\mathbf{p}$ in the $j$:th component to the amount of $k \cdot c$ for some suitable constant $c$.

3. Training the Model

To try out this scheme, the CANDIDE-3 model has been (manually) adapted to 26 images of one person from different angles and with different facial expressions. Since there is only one person in the training data set, the shape parameters $\mathbf{\sigma}$ need not to be changed, and thus only the global motion parameter vector $\mathbf{\gamma}$ and the animation parameter vector $\mathbf{\alpha}$ has been changed.

As Animation Units, the following Action Units from CANDIDE-3 have been chosen:

1. Jaw drop
2. Lip stretcher
3. Lip corner depressor
4. Upper lip raiser
5. Eyebrow lowerer
6. Outer eyebrow raiser
The adapted model has for each image been normalized to a standard shape with the size 40x44 pixels, see Fig. 1, and the resulting training textures collected in a matrix. A PCA has been performed on this matrix, to compute the Texture Units. The mean texture \( \overline{x} \) has been subtracted from the training textures prior to the PCA, but the DC-level has not.

With the TUs available, all the parameters have been perturbed, one by one and for each image, in steps of 0.01 in the range \([-0.1, 0.1]\), and the matrix \( R \) estimated. From \( R \), the update matrix \( U \) has been computed.

4. Testing

The algorithm has been implemented in Visual C++ 6. For image processing and face modelling tasks, the ImgProc Library and the FaceModel Library developed within the InterFace project have been used. The ImgProc Library is based on IPL from the Intel Performace Library Suite. OpenGL has been used for texture mapping and visualization, thus moving the texture mapping computations to the graphics hardware. The images have been captured using a Sony EVI-D31 camera and digitized with an Asus V3800 Ultra graphics card with video input. The tests has been performed on a PC with a 500 MHz Pentium III processor.

In each iteration, four values of \( k \) are tested (see (11)). Each iteration, optimizing 12 parameters, takes about 50 ms, and typically 5 iterations are needed between two frames in the video sequence, making the tracking run with a framerate of about 3 Hz. If the time for each iteration could be reduced somewhat, the framerate would be higher. Thus fewer iterations would be needed each frame (since the estimate from the last frame would fit better), which would improve the framerate even more. We aim at a framerate of 15 Hz using the current hardware.
Figure 3. The model adapted to three frames of a video sequence.
Three frames from a video sequence to which the model has been adapted is shown in Fig. 3. As can be seen, the global adaptation (rotation, scale, translation) is good and the eyebrow parameters behave well. However, the mouth is often opened a bit too much. All parameters suffer from the algorithm being greedy and easily getting stuck in a local optimum. Typically, this results in the model not being able to follow fast moves, like when the mouth is closed too fast or when the head is moved away quickly, as in Fig. 2.

The video sequence contains the same person as in the training data, however, no glasses are worn in the training data and the lighting and shaving conditions are somewhat different. Those small changes are handled quite well, but to track other persons, the Shape Units should be activated and more training data collected.

5. Future work

This is our first experiment on the AAA, and there are numerous ways this scheme can be improved. There are also several possible improvements that should be implemented and evaluated.

- The Texture Units are colour images (RGB). A conversion to grayscale would make the AAA update step three times faster (modulo the time for converting the input image vector from RGB to grayscale). It should also be investigated how this influences the performance.
- The DC-level is not subtracted from the TU s (and thus not from the input texture). Doing this should improve robustness to varying lighting conditions.
- The Animation Units are chosen quite ad-hoc.
- No Shape Units at all are used.
- The training data set is very small.
- The AAA drives only the geometry parameters, instead appearance parameters. Those two approaches have been compared by Cootes et al [1], showing that the current approach may be faster but less accurate.
- A profiling of the program should be performed, finding the bottlenecks and enabling optimization.
- A multi-scale search should be implemented, to decrease the computation time and to make the need for a good initial estimate smaller.
- The AAA should be combined with an algorithm that quickly finds a rough estimate of the face location.
- When used for tracking in a video sequence, the initial estimate in a frame should be better predicted than just the adaptation from the previous frame. This could be done, for example, with a simple motion estimation in a few points, a Kalman filter, or a combination thereof.
- The shape parameters should be Kalman-filtered (with a stationary model) to make them quickly converge to a stable value.
6. Conclusion

The described approach looks promising, with many improvements within reach. It is our belief that a real-time model-based coding system should be possible to build on this foundation.

Acknowledgements

This work is sponsored by the Swedish SSF-project VISIT and the European IST-project InterFace. The author is greatly thankful to Dr. Tim Cootes at the University of Manchester for the simple method of computing the update matrix.

References


