Active Appearance Models for Face Recognition

Paul Ivan
ivan.paul@gmail.com
Supervisor: dr. Sandjai Bhulai

April 4, 2007

Vrije Universiteit Amsterdam
Faculteit der Exacte Wetenschappen
Business Mathematics & Informatics
De Boelelaan 1081a
1081 HV Amsterdam

Abstract

A growing number of applications are starting to use face recognition as the initial step towards interpreting human actions, intention, and behaviour, as a central part of next-generation smart environments. Recognition of facial expressions is an important example of face-recognition techniques used in these smart environments. In order to be able to recognize faces, there are some difficulties to overcome. Faces are highly variable, deformable objects, and can have very different appearances in images depending on pose, lighting, expression, and the identity of the person. Besides that, face images can have different backgrounds, differences in image resolution, contrast, brightness, sharpness, and colour balance.

This paper describes a model-based approach, called Active Appearance Models, for the interpretation of face images, capable of overcoming these difficulties. This method is capable of 'explaining' the appearance of a face in terms of a compact set of model parameters. Once derived, this model gives the opportunity for various applications to use it for further investigations of the modelled face (like characterise the pose, expression, or identity of a face). The second part of this paper describes some variations on Active Appearance Models aimed at increasing the performance and the computational speed of Active Appearance Models.

Acknowledgements

This paper was written as part of the master Business Mathematics and Informatics at the Vrije Universiteit, Amsterdam. The main goal of this assignment is to write a clear and concise paper on a certain scientific problem, with a knowledgable manager as the target audience.

I want to thank dr. Sandjai Bhulai for helping me defining a good subject for this paper and his comments during the writing-process.

Paul Ivan
Amsterdam, April 4, 2007

Contents

Chapter 1

Introduction

Researchers today are actively building smart environments. These environments, such as rooms, cars, offices, and stores, are equipped with smart visual, audio, and touch sensitive applications. The key goal of these applications is usually to give machines perceptual abilities that allow them to function naturally with people, in other words, to recognize the people and remember their preferences and characteristics, to know what they are looking at, and to interpret their words, gestures, and unconscious cues such as vocal prosody and body language [7].

A growing number of applications are starting to use face recognition as the initial step towards interpreting human actions, intention, and behaviour, as a central part of next-generation smart environments. Many of the actions and behaviours humans display can only be interpreted if you also know the person's identity, and the identity of the people around them.

Recognition of facial expressions is an important example of face-recognition techniques used in these smart environments. It can, for example, be useful for a smart system to know whether the user looks impatient because information is being presented too slowly, or confused because it is going too fast. Facial expressions provide clues for identifying and distinguishing between these different moods. In recent years, much effort has been put into the area of recognizing facial expressions, a capability that is critical for a variety of human-machine interfaces, with the hope of creating person-independent expression recognition capability. Other examples of face-recognition techniques are recognizing the identity of a face/person or characterizing the pose of a face.

Various fields could benefit of systems capable of automatically extracting this kind of information from images (or sequences of images, like a videostream). For example, a store equipped with a smart system capable of expression recognition could benefit from this information in several ways.

Such a system could monitor the reaction of people to certain advertisements or products in the store, or the other way around, they could adjust their in-store advertisements based on the expressions of the customers. In the same manner, marketing research could be done with cameras monitoring the reaction of people to their products. Face recognition techniques aimed at recognizing the identity of a person, could help such a store when a valued repeat customer enters a store.

Other examples are, behaviour monitoring in an eldercare or childcare facility, and command-and-control interfaces in a military or industrial setting. In each of these applications identity information is crucial in order to provide machines with the background knowledge needed to interpret measurements and observations of human actions.

Goals and Overview In order to be able to recognize faces, there are some difficulties to overcome. Faces are highly variable, deformable objects, and can have very different appearances in images depending on pose, lighting, expression, and the identity of the person. Besides that, face images can have different backgrounds, differences in image resolution, contrast, brightness, sharpness, and colour balance. This means that interpretation of such images/faces requires the ability to understand this variability in order to extract useful information and this extracted information must be of some manageable size, because a typical face image is far too large to use for any classification task directly.

Another important feature of face-recognition techniques is real-time applicability. For an application in a store, as described above, to be successful, the system must be fast enough to capture all the relevant information derived from video images. If the computation takes too long, the person might be gone, or might have a different expression. The need for real-time applicability thus demands for high performance and efficiency of applications for face recognition.

This paper describes a model-based approach for the interpretation of face images, capable of overcoming these difficulties. This method is capable of ‘explaining’ the appearance of a face in terms of a compact set of model parameters. The created models are realistically looking faces, closely resembling the original face depicted in the face image. Once derived, this model gives the opportunity for various applications to use it for further investigations of the modelled face (like characterise the pose, expression, or identity of a face).

This method, called Active Appearance Models, in its basic form is described in Chapter 2. Because of the need for real-time applications using this technology, variations on the basic form aimed at increasing the performance and the computational speed are discussed in Chapter 3. Some experimental results of comparative tests between the basic form and the variations are presented in Chapter 4. Finally, a general conclusion/discussion will be given in Chapter 5.

Chapter 2

Active Appearance Models

The Active Appearance Model, as described by Cootes, Taylor, and Edwards (see, [1] and [6]) requires a combination of statistical shape and texture models to form a combined appearance model. This combined appearance model is then trained with a set of example images. After training the model, new images can be interpreted using the Active Appearance Search Algorithm. This chapter will describe these models in detail, mostly following to the work of [1], [6], and [5].

2.1 Statistical Shape Models

The statistical shape model is used to represent objects in images. A shape is described by a set of n points. The points are called landmarks and are often in 2D or 3D space. The goal of the statistical shape model is to derive a model which allows us to both analyze new shapes and to synthesize shapes similar to those in the training set. The training set is often generated by hand annotation of a set of training images, an example of such a hand-annotated image can be seen in Figure 2.1. By analyzing the variations in shape over the training set, a model is built which can mimic this variation.

If in the two dimensional case a shape is defined by n points, we represent the shape by a $2n$ element vector formed by concatenating the elements of the individual point positions:

$$x = (x_1, y_1, x_2, y_2, \ldots, x_n, y_n). \tag{2.1}$$

If we have a training set of s training examples, we generate s such vectors x^i, in which x^i is the shape vector of shape i. Now, because faces in the images in the training set can be at different positions, of different size, and have different orientation, we wish to align the training set before we perform

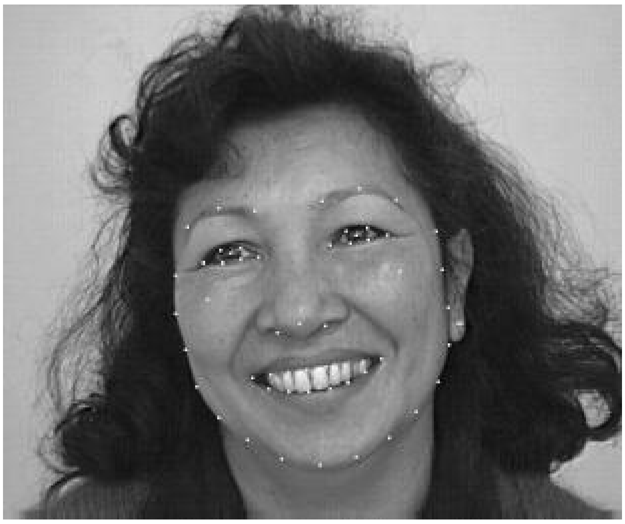

Figure 2.1: Hand annotated face

statistical analysis. The most popular approach of doing this is aligning each shape by minimizing the sum of distances of each shape to the mean shape vector, $\bar{x}$, over all s shape vectors.

$$D = \sum_{i=1}^{s} ||x^i - \bar{x}||^2. \tag{2.2}$$

This alignment can be done by applying re-positioning, scaling, and rotation of the shape, which are valid operations considering the invariability of shapes under Euclidean transformations. Although useful, this method is poorly defined unless there are clearly defined constraints of the alignment of the mean shape, like ensuring it is centered around the origin, has unit size, and some fixed orientation. Cootes and Taylor describe a simple iterative approach for applying this alignment.

1. Translate each example so that its center of gravity[1] is at the origin.

2. Choose one example as an initial estimate of the mean shape and scale so that $||\bar{x}|| = 1$.[2]

[1] The point in any solid where a single applied force could support it; the point where the mass of the object is equally balanced. The center of gravity is also called the center of mass.

[2] $||x||$ is defined as the norm of the n-dimensional vector x and can be calculated by $||x|| = \sqrt{x_1^2 + x_2^2 + \ldots + x_n^2}$

3. Record the first estimate as $\bar{x}^i$, with $i = 0$ to define the default reference frame.

4. Align all the shapes with the current estimate of the mean shape.

5. Re-estimate the mean from the aligned shapes.

6. Apply constraints in the current estimate of the mean by aligning it with $\bar{x}^i$ and scaling so that $||\bar{x}^{i+1}|| = 1$, set $i = i + 1$ and record this estimate as $\bar{x}^i$.

7. If not converged, return to 4. (convergence is declared if the estimate of the mean does not change significantly after an iteration.)

We now have a set s of points x^i, aligned into a common co-ordinate frame. These vectors form a distribution in the $2n$ dimensional space in which they live. We wish to model this distribution to be able to generate new examples, similar to those in the training set, and to be able to examine new shapes to decide whether they are plausible examples.

We would like to have a parametrized model M of the form $x = M(b)$, where b is a vector of the parameters of the model. To be able to derive such a model we first reduce the dimensionality of the data from $2n$ to a more manageable size. This is done by applying Principal Component Analysis (PCA). PCA is used to extract the main features of the data, by seeking the direction in the feature space which accounts for the largest amount of variance in the data set with possible correlations between variables. This direction (the first principal component) becomes the first axis of the new feature space. This process is repeated to derive the second principal component, and so on until either all variance is explained in the new feature space or the total explained variance is above a certain threshold (l). This approach is as follows:

1. Compute the mean of the data,

$$\bar{x} = \frac{1}{s}\sum_{i=1}^{s} x^i. \tag{2.3}$$

2. Compute the sample covariance of the data[3],

$$S = \frac{1}{s-1}\sum_{i=1}^{s}(x^i - \bar{x})(x^i - \bar{x})^T. \tag{2.4}$$

[3] Note that, since x^i is a vector, this matrix can be seen as the covariance matrix between the individual (scalar) elements x_i of the vector x^i.

3. Compute the eigenvectors ϕ_i and the corresponding eigenvalues $\lambda_{s,i}$ of S (sorted such that $\lambda_{s,i} \geq \lambda_{s,i+1}$).

Then, if P_s contains the l eigenvectors corresponding to the largest eigenvalues, then we can approximate any shape vector x of the training set using:

$$x \approx \bar{x} + P_s b_s, \tag{2.5}$$

where $P_s = (\phi_1 | \phi_2 | \ldots | \phi_l)$ is an orthogonal matrix (thus $P_s^T = P_s^{-1}$ and $P_s^T P_s = I_l$) and b_s is an l-dimensional vector given by

$$b_s = P_s^T (x - \bar{x}). \tag{2.6}$$

Now we have the parametrized form, in which the vector b_s defines the set of parameters of the model. By the use of Principal Component Analysis we have reduced the number of parameters from s to l with $l < s$. Depending on l this can be a significant reduction in dimensionality. By varying the elements of b we can vary the shape. The variance of the i^{th} parameter b_i across the training set is given by $\lambda_{s,i}$. By applying limits of $\pm 3\sqrt{\lambda_{s,i}}$ to the parameters of b_s we ensure that the shape generated is similar to those in the original training set. The number of parameters in b_s is defined as the number of modes of variation of the shape model.

2.2 Statistical Texture Models

To be able to synthesize a complete image of an object we would like to include the texture information of an image. By 'texture' we mean, the pattern of intensities or colours across an image patch.

Given an annotated training set, we can generate a statistical model of shape variation from the points. Given a mean shape, we can warp each training image into the mean shape, to obtain a 'shape-free' patch. From that we can build a statistical model of the texture variation in this patch. Warping each training image means, changing an image so that its control points match the mean shape (using a triangulation algorithm, see Appendix F of [6]). This is done to remove spurious texture variations due to shape differences. We then sample the intensity information from the shape-normalized image over the region covered by the mean shape to form a texture vector, g^{image}.

To minimize the effect of global lighting, the shape-free patches should be photometrically aligned, or in other words, the shape-free patches should be normalized. This is done by minimizing the sum of squared distances E_g between each texture vector and the mean of the aligned vectors $\bar{g}$, using offsetting (changing brightness) and scaling (changing the contrast) of the entire shape-free patch:

$$E_g = \sum_{i=1}^{s} |g^i - \bar{g}|^2, \tag{2.7}$$

where s is the number of shape vectors and texture vectors and thus the number of images in the training set.

E_g is minimized using the transformation $g^i = (g^{image} - \beta 1)/\alpha$, where α is the scaling factor and β is the offset.

$$\alpha = g^{image} \cdot \bar{g}, \qquad \beta = (g^{image} \cdot 1)/n, \tag{2.8}$$

where n is the number of elements in the vector.

Obtaining the mean of the normalized data is a recursive process, as the normalization is defined in terms of the mean. This can be solved by an iterative algorithm. Use one of the examples as the first estimate of the mean, align the others to it (using 2.7 and 2.8) and re-estimate the mean, calculate E_g and keep iterating between the two until E_g is converged (does not get smaller anymore).

The next step is to apply PCA to the normalized data, in a similar manner as with the shape models. This results in:

$$g \approx \bar{g} + P_g b_g, \tag{2.9}$$

in which P_g contains the k eigenvectors corresponding to the largest eigenvalues $\lambda_{g,i}$ and b_g are the grey-level parameters of the model. The number of parameters are called the number of texture modes.

The elements of b_i are again bound by:

$$-3\sqrt{\lambda_{g,i}} \leq b_i \leq 3\sqrt{\lambda_{g,i}}. \tag{2.10}$$

If we represent the normalization parameters α and β in a vector $u = (\alpha - 1, \beta)^T$, we represent u as $u = (u_1, u_2)^T$, and $g = (g^{image} - \beta 1)/\alpha$, we can state that the transformation from g to g^{image} is the following:

$$T_u(g) = (1 + u_1)g + u_2 1. \tag{2.11}$$

Now we can generate the texture in the image in the following manner:

$$g^{image} \approx T_u(\bar{g} + P_g b_g) = (1 + u_1)(\bar{g} + P_g b_g) + u_2 1. \tag{2.12}$$

2.3 The Combined Appearance Model

The appearance model combines both the shape model and the texture model. It does this by combining the parameter vectors b_s and b_g to form a combined parameter vector b_{sg}. Because these vectors are of a different nature and thus of a different relevance, one of them will be weighted.

$$b_{sg} = \begin{pmatrix} W_s b_s \\ b_g \end{pmatrix}. \tag{2.13}$$

Since b_s has units of distance and b_g has units of intensity, they cannot be compared directly. To make b_s and b_g commensurate, the effect of varying b_g on the sample g must be estimated. This can be done by systematically displacing each element of b_s from its optimal value on each training example and calculating the corresponding difference in pixel intensities.

A simpler alternative is to set $W_s = rI$ where r^2 is the ratio of the total intensity variation to the shape variation (in normalized frames). Note that we already calculated the intensity variation and the shape variation in the form of the eigenvalues $\lambda_{s,i}$ and $\lambda_{g,i}$, of the covariation matrix of the shape vectors and the intensity vectors. Thus:

$$W_s = \frac{\lambda_g^+}{\lambda_s^+}, \tag{2.14}$$

with,

$$\lambda_g^+ = \sum_{i=1}^{k}(\lambda_{g,i}), \qquad \lambda_s^+ = \sum_{i=1}^{l}(\lambda_{s,i}). \tag{2.15}$$

where, $\lambda_{s,i}$ are the l eigenvalues of the covariance matrix of the shape vector and $\lambda_{g,i}$ are the k eigenvalues of the covariance matrix of the texture vector.

PCA is once more applied to these vectors, giving the final model:

$$b_{sg} = P_c c, \tag{2.16}$$

where P_c are the eigenvectors belonging to the m largest eigenvalues of the covariance matrix of combined and weighted texture- and shape modes b_{sg}. The vector c is a vector of appearance parameters controlling both the shape and grey-levels of the model, defined as the Appearance modes of variation. Note that the dimension of the vector c is smaller since $m \leq l + k$. Now from this model we can extract an approximation of the original shape and texture information by calculating:

$$x = \bar{x} + P_s W_s^{-1} P_{cs} c, \qquad g = \bar{g} + P_g P_{cg} c, \tag{2.17}$$

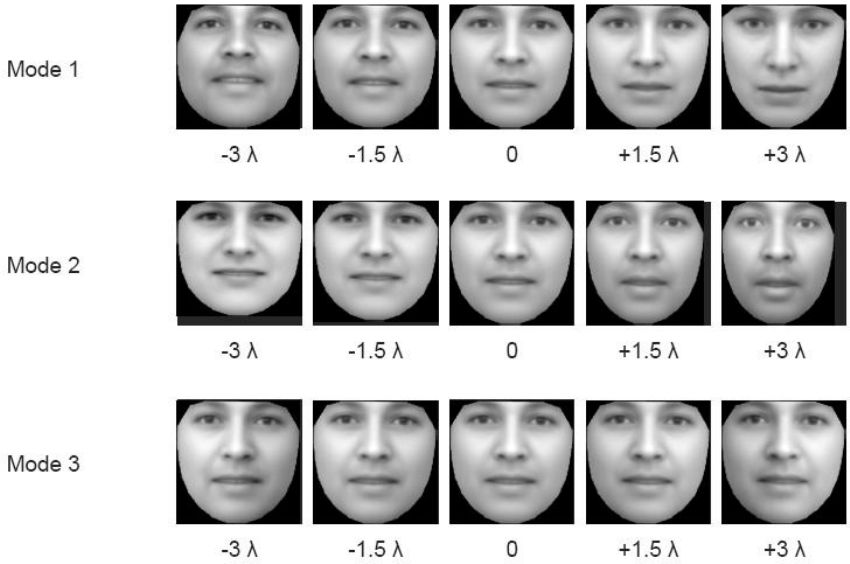

Figure 2.2: Result of varying the first three appearance modes

with,

$$P_c = \begin{pmatrix} P_{cs} \\ P_{cg} \end{pmatrix}. \tag{2.18}$$

From an image, we can now extract a compact vector c which describes the appearance (both shape and texture) of the depicted face. And vice versa, given a vector c, an image can be synthesized by calculating a shape-free patch from b_g and warping this to the shape described by b_s. The elements of the appearance vector c are referred to as the appearance modes.

Figure 2.2 (taken from [5]) shows the effect of varying the first three (the most significant) appearance modes. Note that the image in the middle, at zero, is the mean face (derived from a particular training set). From this image, we can clearly see how the first two modes affect both the shape and the texture information of the face model. Note that, the composition of the training set, the amount of variance retained in each step and the weighting of shape versus texture information will determine the most significant appearance modes and what these modes look like (or what their influence is).

2.4 The Active Appearance Search Algorithm

Until now we have discussed the training phase of the appearance model. In this section the Active Appearance Search Algorithm will be discussed. This algorithm allows us to find the parameters of the model, which generate a synthetic image as close as possible to a particular target image, assuming a reasonable starting approximation[4].

Interpretation of a previously unseen image is seen as an optimization problem in which the difference between this new image and the model (synthesized) image is minimized.

$$\delta I = I_i - I_m, \tag{2.19}$$

where, I_i is the vector of grey-level values in the image and I_m is the vector of grey-level values for the current model parameters. We wish to minimize $\Delta = |\delta I|^2$, by varying the model parameters, c. This appears to be a difficult high-dimensional optimization problem, but in [1] Cootes et al. pose that the optimal parameter update can be estimated from δI. The spatial pattern in δI, encodes information about how the model parameters should be changed in order to achieve a better fit. There are basically two parts to the problem:

1. Learning the relationship between δI and the error in the model parameters δc,
2. Using this knowledge in an iterative algorithm for minimizing Δ.

The appearance model has one compact parameter vector c, which controls the shape and the texture (in the model frame) according to:

$$x = \bar{x} + Q_s c, \qquad g = \bar{g} + Q_g c, \tag{2.20}$$

where

$$Q_s = P_s W_s^{-1} P_{cs}, \qquad Q_g = P_g P_{cg}, \tag{2.21}$$

where $\bar{x}$ is the mean shape and $\bar{g}$ is the mean texture in a mean-shaped patch.

A shape in the image frame, X, can be generated by applying a suitable transformation to the point, $x : X = S_t(x)$. Valid transformations are, scaling (s), an in-plane rotation (θ), and a translation (l_x, l_y). If for linearity we represent the scaling and rotation as (s_x, s_y) where $s_x = (s\cos\theta - 1)$ and

[4] To find a reasonable starting position, often a separate module/application is used, which has a fast way of finding an estimate of the position of a face in an image ([5, p.9])

$s_y = s \sin\theta$, then the pose parameter vector $t = (s_x, s_y, l_x, l_y)^T$ is zero for the identity transformation and $S_{t+\delta t(x)} \approx S_t(S_{\delta t}(x))$. Now, in homogeneous co-ordinates, t corresponds to the transformation matrix:

$$S_t = \begin{pmatrix} 1+s_x & -s_y & l_x \\ s_y & 1+s_x & l_y \\ 0 & 0 & 1 \end{pmatrix}. \tag{2.22}$$

For the AAM we must represent small changes in pose using a vector, δt. This is to allow us to predict small pose changes using a linear regression model of the form $\delta t = Rg$. For linearity the zero vector should indicate no change, and the pose change should be approximately linear in the vector parameters. This is satisfied by the above parameterization. The AAM algorithm requires us to find the pose parameters t' of the transformation obtained by first applying the small change given by δt, then the pose transform given by t. Thus, find t' so that $S_{t'}(x) = S_t(S_{\delta t}(x))$. Now it can be shown that for small changes, $S_{\delta t_1}(S_{\delta t_2}(x)) \approx S_{(\delta t_1 + \delta t_2)}(x)$, see Appendix D of [6].

From the appearance model parameters c and shape transformation parameters, t, we get the position of the model points in the image frame X. This gives the shape of the image patch to be represented by the model. During the matching phase we sample the pixels in this region of the image, g^{image}, and project into the texture model frame, $g_s = T_u^{-1}(g^{image})$, with T_u from (2.12). Then, the current model texture is given by $g_m = \bar{g} + Q_q c$. The current difference between model and image in the normalized texture frame is then:

$$r(p) = g^s - g^m, \tag{2.23}$$

where p are the parameters of the model, $p^T = (c^T|t^T|u^T)$. A scalar measure of difference is the sum of squares of elements of r, $E(p) = r(p)^T r(p)$. A first order Taylor expansion of (2.23) gives,

$$r(p + \delta p) = r(p) + \frac{\partial r}{\partial p}\delta p, \tag{2.24}$$

where the ij^{th} element of matrix $\frac{\partial r}{\partial p}$ is $\frac{dr_i}{dp_j}$.

Suppose during matching our current residual is r. We wish to choose δp so as to minimize $|r(p+\delta p)|^2$. By equating (2.24) to zero we obtain the RMS (root mean squared) solution.

$$\delta p = -Rr(p), whereR = (\frac{\partial r}{\partial p}^T \frac{\partial r}{\partial p})^{-1} \frac{\partial r}{\partial p}^T. \tag{2.25}$$

Normally it would be necessary to recalculate $\frac{\partial r}{\partial p}$ at every step, an expensive operation. However, we assume that since it is being computed in a normalized reference frame, it can be considered approximately fixed. We can thus estimate it once from our training set, We estimate $\frac{\partial r}{\partial p}$ by numeric differentiation, systematically displacing each parameter from the known optimal value on typical images and computing an average over the training set. Residuals at displacements of differing magnitudes are measured (typically up to 0.5 standard deviations of each parameter) and combined with a Gaussian kernel to smooth them. We then precompute R and use it in all subsequent searches with the model.

Now if we have computed the matrix R, we can construct an iterative method for solving the optimization problem. Given a current estimate of model parameters, c, the pose t, the texture transformation u, and the image sample at the current estimate g^{image}, one step of the iterative matching procedure is as follows:

1. Project the texture sample into the texture model frame using $g^s = T_u^{-1}(g^{image})$.
2. Evaluate the error vector, $r(p) = g^s - g^m$, and the current error, $E = |r(p)|^2$.
3. Compute the predicted displacements, $\delta p = -Rr(p)$.
4. Update the model parameters $\hat{p} \to p + k\delta p$, where initially $k = 1$.
5. Calculate the new points, $\widehat{X}$ and the model frame texture $\hat{g}^m$.
6. Sample the image at the new points to obtain $\hat{g}^{image}$.
7. Calculate a new error vector, $r(\hat{p}) = T_{\hat{u}}^{-1}(\hat{g}^{image}) - \hat{g}^m$.
8. If $|r(\hat{p})|^2 < E$ then accept the new estimate (record $p = \hat{p}$), otherwise try at $k = 0.5, k = 0.25$, etc.
9. Repeat this procedure until no improvement is made to the error, $|r(p)|^2$, and convergence is declared.

2.5 Multi-resolution Implementation

Cootes and Taylor ([6]) propose a more efficient implementation of the previously described iterative algorithm. They use a multi-resolution implementation, in which they iterate to convergence at each level. The idea comes from the multi-resolution Active Shape Models. The method involves first searching and matching in a coarse image and then further matching in a series of finer resolution images.

For all the images in the training and test set, a Gaussian image pyramid is built. This pyramid represents the different resolution levels, in which level 0 is the original image. The next level (level 1) is formed by smoothing the original image and then sub-sampling to obtain an image with half the number of pixels in each dimension, and so on for subsequent levels.

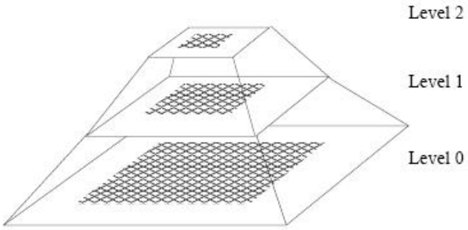

Figure 2.3: A Gaussian image pyramid is formed by repeated smoothing and sub-sampling

During training different models for different resolution levels are built. During search, the iterative algorithm from Section 2.4 is performed at the different resolution levels. Starting with the highest level and iterated until convergence is declared at that level.

2.6 Example of a Run

The previous sections described the whole process underlying the Active Appearance Models. In this section a single run is presented. When an image is presented to the system, first an approximation of the position of a face (see Figure 2.4) is found. In the next stage, a model fit is created as depicted in Figure 2.5.

From the image in Figure 2.5 and 2.4, we can see that all the main characteristics of the face are preserved reasonably well and at first glance the model and original might actually be confused. When we look closer, we can see that the skin has become smoother, edges are less sharp and minor skin blemishes have largely disappeared. It should be noted that this is totally due to the variation in the training set. If the characteristics of a face image presented to the system deviate greatly from the training set, the fit quality degrades, and vice versa. Figure 2.6 shows the progress of a multi-resolution

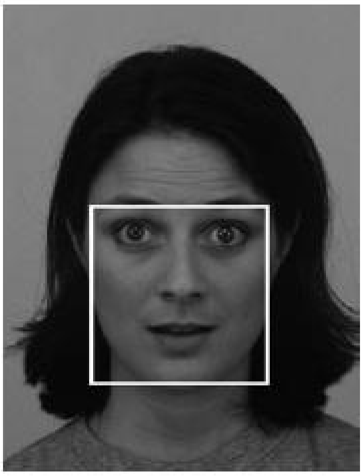

Figure 2.4: Approximation of the position of the face

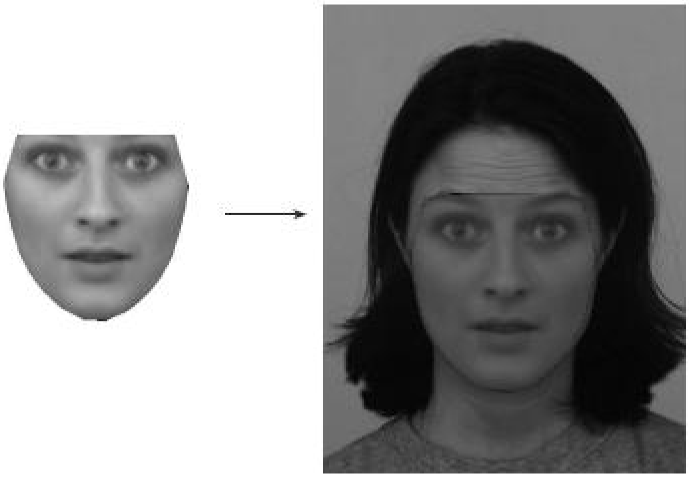

Figure 2.5: Model fit of the face

search. Each starting with the mean model displaced from the true face center.

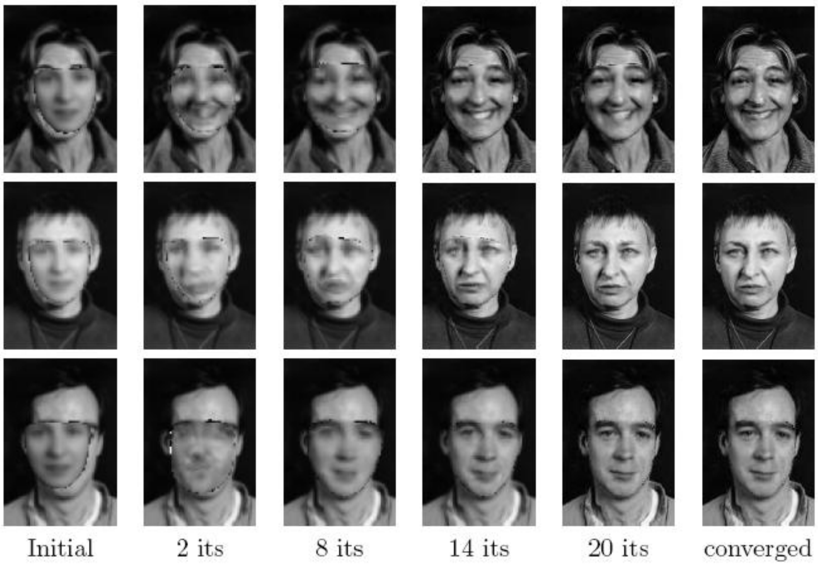

Figure 2.6: Progress of a multi-resolution search

Chapter 3

Variations on the AAMs

Since performance and efficiency is very important when implementing Active Appearance Models (AAMs) in real-time applications, this section describes modifications to the basic AAM search algorithm aimed at improving the speed and robustness of search.

3.1 Sub-sampling during Search

In [2], Cootes et al. describe that during search, the basic AAM formulation samples all the points in the model to obtain g^s. There may be more than 10,000 pixels, but fewer than 100 parameters. There is thus considerable redundancy. This indicates that it might be possible to obtain good results by only sampling the most important (a sub-set) of the modelled pixels. During the training phase the update matrix R is computed, this matrix is used to calculate the change in the i^{th} parameter, δc_i :

$$\delta c_i = a_i \delta g, \tag{3.1}$$

where a_i is the i^{th} row of R. The elements of a_i indicate the significance of the corresponding pixel in the calculation of the change in the parameter. From this we can derive what pixels are most important for a certain parameter. By sorting the elements of a_i by absolute value and selecting the largest, we can choose the most useful sub-set for a given parameter. However, the pixels which best predict changes to one parameter, may be less or not useful for a different parameter.

We can solve this by computing the best $u\%$ of elements for each parameter and then generate the union of such sets. If u is small enough, the union will be less than all the elements. With this sub-set we perform a new multivariate regression, to compute the relationship a' between the changes in the sub-set of samples, $\delta g'$, and the changes in the parameters

$$\delta c = a' \delta g', \tag{3.2}$$

and using the same search algorithm, but using only a sub-set of all the pixels.

3.2 Search Using Shape Parameters

In the original formulation the parameters c are manipulated. A different approach is to use image residuals to drive the shape parameters b_s and computing the texture parameters b_g, and thus c directly from the image given the current shape. This might be useful when there are few shape modes and many texture modes. This approach gives a different update function:

$$\delta b_s = B \delta g. \tag{3.3}$$

Now δg is given by the difference between the current image sample g^s and the best fit of the grey-level model to it, g^m.

$$\delta g = g^s - g^m = g^s - (\bar{g} + P_g b_g), \tag{3.4}$$

where $b_g = P_g^T (g^s - \bar{g})$.

We now have to learn the relationship (B) between δb_s and δg. We know that any δg is orthogonal to the columns of P_g, thus the update equation becomes:

$$\delta b_s = B(g^s - \bar{g}) = B g^s - b_{offset}. \tag{3.5}$$

where, $b_{offset} = B\bar{g}$ and thus a constant offset to the parameters b_s. Fitting a model to an image now simplifies to keeping track of the pose t, the texture transformation u, and shape parameters b_s. The texture parameters can be computed directly from the sample at the current shape. So in this case $p = (b_s^T | t^T | u^T)$. In a training phase one learns the relationships:

$$\delta p = -Rr(p), \tag{3.6}$$

where R is calculated in the same manner as in Equation 2.25, except for the fact that the model parameters p do not consist of the texture parameters (b_g).

During the search the update step is modified as follows:

1. Project the texture sample into the texture model frame using $g^s = T_u^{-1}(g^{image})$.
2. Fit the texture model to the sample using $b_g = P_g^T(g^s - \bar{g})$.
3. Compute the residual[1] as $r(p) = g^s - g^m$.
4. Predict the pose parameters, the texture transformation, and the shape parameter updates using 3.6.
5. Apply and test the updates as for the basic algorithm.

We can apply the constraints of the combined appearance model by computing c, applying the constraints and then recomputing the shape parameters. The magnitude of the residual $||r(p)||$ can be used to test for convergence.

This method may be faster in cases where there are significantly fewer modes of shape variation than the combined appearance modes. However, it might perform less fast, because it is only indirectly driving the parameters controlling the full appearance c.

3.3 Direct AAMs

Hou et al. ([4]) suggest that in some cases it is possible to predict the shape directly from the texture, which leads to a faster algorithm.

If we recall from Chapter 2 that:

$$x = \bar{x} + P_s b_s, \qquad g = \bar{g} + P_g b_g. \tag{3.7}$$

And recall that we denoted the individual parameter vectors of the shape and the texture model as:

$$b_s = P_{cs}c, \qquad b_g = P_{cg}c. \tag{3.8}$$

Now, Hou et al. claim that the relationship between the texture and shape information, for faces, is many to one.

$$b_s = Sb_g = P_{cs}P_{cg}^{-1}b_g, \tag{3.9}$$

where P_{cg}^{-1} is the pseudo-inverse of P_{cg}. If the rank of P_{cg} is larger than the number of shape parameters, we can accurately predict the shape parameters from the texture. They suggest the following iterative procedure:

[1] Note that the residual r is denoted as a function of p, since g^s and g^m both depend on parts of the current model parameters p.

1. Fit the texture model to the normalized sample using $b_g = P_g^T(g_s - \bar{g})$,
2. Compute the residual as $r = g_s - g_m$,
3. Predict the pose using $\delta t = R_t r$,
4. Apply and test the updates as in the basic algorithm.

3.4 Compositional Approach

According to Baker and Matthews ([8]), the essentially additive method of updating the parameters in the basic form could be a problem. They propose an alternative compositional updating scheme. Their method applies to the case in which separate (independent) shape and texture models are used. Then, the shape model equation can be thought of as a parametrized transformation of the mean shape.

$$x = T_{b_s}(\bar{x}). \tag{3.10}$$

where,

$$T_{b_s}(\bar{x}) = \bar{x} + P_s b_s. \tag{3.11}$$

When we use an additive update of the form $b_s \rightarrow b_s + \delta$, it leads to a new transformation

$$x_{new} = T_{b_{s+\delta}}(\bar{x}). \tag{3.12}$$

However, it might be more natural to think of the update itself as a transformation,

$$x_{new} = T_{b_s}(T_\delta(\bar{x})). \tag{3.13}$$

A way of achieving this is to approximate the transformation using thin-plate splines ([8]) as follows

1. Compute the thin plate spine, $T_{tps}(.)$ which maps the point $\hat{x}$ to x,
2. Compute the modified mean points, $x_\delta = \hat{x} + P_s\delta$,
3. Apply the transformation, $x' = T_{tps}(x_\delta)$,
4. Find the shape parameters which best match, $b'_s = P_s^T(x' - \hat{x})$.

The sampling and updating is otherwise identical to that for the shape AAM described above.

Chapter 4

Experimental Results

This chapter will show the results from some comparative experiments.

4.1 Sub-sampling vs. Shape vs. Basic

In [2], Cootes et al. describe the results of a comparative experiment. To compare the variations described above (Sub-sampling and Shape), an appearance model was trained on a set of 300 labelled images of faces. The set contains several images of 40 people each, with a variety of different expressions. Each image was hand-annotated with 122 landmark points. The following table summarizes the number of parameters (modes of variation) of the different parts of the model:

Model	Modes of variation
Shape model	36
Texture model (10000 pixels)	223
Combined model	93

Table 4.1: Overview of model parameters

Three versions of the AAM were trained for these models. A Basic AAM, a Sub-sampling AAM using a sub-set of 25% of the pixels to drive the parameters c, and a Shape AAM trained to drive the shape parameters b_s alone.

A test set of 100 unseen new images (of the same people but with different expressions) was used to test the performance of the algorithms. From the hand-annotated landmarks the optimal pose was found, and the model was displaced by (+15, 0, -15) pixels in the x and y direction. They applied a multi-resolution search with 9 tests per image. Two different search regimes were used:

1. A maximum of 5 iterations allowed at each resolution level. Each iteration tested the model at $c \rightarrow c - k\delta c$ for $k = 1.0, 0.5^1, \ldots, 0.5^4$, where the first iteration that gave an improved result was accepted or convergence was declared if none did.

2. The update $c \rightarrow c - \delta c$ was forced without testing whether it was better or not, applying 5 steps at each resolution level.

They used two ways of recording the quality of fit,

- The RMS grey-level error per pixel in the normalized frame, $\sqrt{|\delta v|^2 / n_{pixels}}$,
- The mean distance error per model point.

Some searches fail to converge, this is detected by a threshold on the mean distance error per model point. They considered those searches that have a mean error of > 7.5 to have failed to converge. The results are summarized in Table 4.2.

AAM Type	Search Regime	Failure Rate	Final Errors		Mean
			Point ±0.05	Grey ±0.05	Time ms
Basic	1	4.1%	4.2	0.45	3270
Basic	2	4.6%	4.4	0.46	2490
Sub-Sampling	1	13.9%	4.6	0.60	920
Sub-Sampling	2	22.9%	4.8	0.63	630
Shape	1	11.4%	4.0	0.85	560
Shape	2	11.9%	4.1	0.86	490

Table 4.2: Comparative performance of different AAMs, given displaced centers

The final errors were averaged over those searches which converged successfully. Overall, we can see that the second search regime decreased the quality of the results, but was faster in every case. Sub-sampling considerably sped up the search, but gave poorer overall results and was more likely to fail. Using the Shape AAM was even faster, but led to more failures than the

Basic AAM. Considering the point errors, it leads to more accurate locations if the search converged correctly, but it increased the error in the grey-level match.

Considering the overall results of this experiment, some questions remain unanswered. At first, since they only used a sub-set of 25% of the total pixels in the Sub-sampling AAM, an interesting experiment would be to find out what would happen with a sub-set of say 50%, or 75%. Since the Sub-sampling AAM had a considerable increase in speed, it would be interesting to know if there is a linear relation between the size of the sub-sample, the failure rate, and the mean running time.

Secondly, they used a test set with images of the same people as in the training set (although, with different expressions). This makes this experiment only applicable to some of the possible applications that use these models for further investigation.

4.2 Comparative performance

In [3], the results of some comparative experiments are presented. The goal was to compare the performance of the Basic, the Direct, the Shape, and the Composition algorithms. An appearance model was constructed from 102 face images, each annotated with 68 landmarks. The following table summarizes the number of parameters (modes of variation) and the retained variation of the different parts of the model:

Model	**Modes of Variation**	**Retained Variation**
Shape model	49	98%
Texture model	73	98%
Combined model	92	99.9%

Table 4.3: Overview of model parameters

A test set of 155 text images of people (without glasses or facial hair) was used. On each test image a search was started with the mean model shape at positions displaced from the known optimal center by ± 10 pixels in the x and y direction. They ran a 3-level multi-resolution AAM (allowing at most 10 iterations per level) and then computed the mean error between the model points and the hand-labelled points, the mean distance of the model points to the appropriate hand-labelled boundaries and the RMS texture error (in the model frame). The results are summarized in Table 4.4.

AAM Type	Pt-Pt Error Pixels		Pt-Crv Error Pixels		Texture Error (Grey-scale)		Time ms
	Median	90%ile	Median	90%ile	Median	90%ile	mean
Basic	11.6	15.6	7.8	10.2	18.9	26.2	172
Direct	9.4	13.6	6.3	9.0	15.5	19.9	172
Shape	9.4	15.1	5.0	8.9	12.8	17.2	292
Composition	9.4	15.4	5.0	9.1	12.8	17.5	332

Table 4.4: Comparative performance of different AAMs, with no forced iterations

The results suggest that based on performance, the Shape and Composition AAMs significantly out-perform the Basic and Direct AAMs, and that the Basic AAM is significantly worse than the direct AAM. Based on computational speed, the Basic and Direct AAMs are clearly faster than the Shape and Composition AAMs.

In a different experiment, Cootes and Taylor, use the same model, but only force one iteration at each resolution and then only accept subsequent steps which improve the results. This approach is tested, because they have found that search performance can be improved by applying the predicted update without testing whether it improves the results or not. It appears to allow for jumps over local minima. The results are summarized in Table 4.5.

AAM Type	Pt-Pt Error Pixels		Pt-Crv Error Pixels		Texture Error (Grey-scale)	
	Median	90%ile	Median	90%ile	Median	90%ile
Basic	8.4	12.4	4.7	7.7	13.8	17.9
Direct	11.2	15.5	7.1	10.2	17.0	21.3
Shape	9.8	15.4	5.1	9.1	12.7	16.7
Composition	9.8	15.6	5.2	9.1	12.8	16.9

Table 4.5: Comparative performance of different AAMs, with one forced iteration

Table 4.5 shows that this strategy slightly improves the performance of the Shape and Composition AAMs, degrades the Direct AAM, but significantly improves that of the Basic AAM. The Basic AAM now leads to significantly better location accuracy than in any of the previous experiment.

Chapter 5

Discussion

Before being able to build advanced applications for face recognition, such as expression recognition, or identity recognition, there are some difficulties with basic face recognition to overcome. Firstly, these difficulties include the large variation in the shape of human faces, the large variation in the appearance of face images and the large dimensionality in a typical face image. Secondly, the need for real-time applicability demands for high performance and efficiency of applications for face recognition.

This paper described a model-based approach, called Active Appearance Models, for the interpretation of face images, capable of overcoming these difficulties. The AAM has a way of modelling the variation in different appearances of faces, by training the model with a training set that has this wide variation embedded. By analyszng the variations over the training set, a model is built which can mimic this variation. The quality of fit of the model is thus directly related to the variation in the training set.

The Active Appearance Model uses several stages in modelling the variation embedded in face images. In the first stage the shape of a face is modelled. The large variation of the large variety in the shape of human faces is addressed in this stage by aligning the hand-annotated images in the training set before statistical analysis is performed. In the second stage the texture of a face is modelled. To remove spurious texture variation due to shape differences we warp each training image into the mean shape, to obtain a 'shape-free' patch. By photometrically aligning the shape-free patches, the effect of the large variation in the appearance of face images is minimized. In the final stage both the shape model and the texture model are combined. It does this by combining the two parameter vectors. Because these are of a different nature and thus of a different relevance, one of them is weighted.

To reduce the dimensionality of the images a technique called Principal Component Analysis is applied. This technique is applied after each stage.

With the result from PCA we can derive a parametrized model of the form $x = M(b)$, where b is a (compact) vector of the parameters of the model. Every stage results in such a model, with the last stage resulting in a combined shape and texture model, with a combined vector of parameters.

Now with the use of the Active Appearance Search Algorithm we can find the parameters of the model, which generate a synthetic image as close as possible to a particular target image, assuming a reasonable starting approximation. This is done by minimizing the difference between an image and the model instance of the image.

Because of the importance of performance and efficiency of face recognition techniques, variations on the basic algorithm were discussed. Since some regions of the model may change little when parameters are varied, we only need to sample the image in regions where significant changes are expected (Sub-sampling). This should reduce the cost of each iteration.

The original formulation manipulates the combined shape and grey-level parameters directly. An alternative approach is to use image residuals to drive the shape parameters, computing the texture parameters directly from the image given the current shape (Direct AAM).

Another possible way of increasing the efficiency of the AAM lies in the way the updates of the parameters are computed. In Section 3.4 a variation of the basic AAM was discussed where, instead of an additive approach, a compositional approach is used to update the parameters of the AAM (Compositional AAM).

Yet another variation comes from the idea that in some cases it is possible to predict the shape directly from the texture, when the two are sufficiently correlated (Shape AAM).

The results of two different experiments were presented. Firstly, the result of a comparative experiment between the Basic, the Direct, the Shape, and the Compositional approach was presented. The three alternatives all outperformed the basic algorithm when a straightforward test was done. However, a minor modification to the search algorithm (allowing one or more 'forced' iterations in which an update is done regardless of whether it improves the error) improved the result of the basic algorithm leading to the basic AAM outperforming all the alternatives when the errors on point location are concerned.

Secondly, the results of a comparative experiment between the Basic, the Shape, and the Sub-sampling AAM were presented. Sub-sampling and driving the shape parameters during search both lead to faster convergence, but were more prone to failure. The Shape AAM was able to locate the points slightly more accurately than the original formulation. And testing for improvement and convergence at each iteration slowed down the search, but lead to better final results.

Bibliography

[1] T.F. Cootes, G.J. Edwards, C.J. Taylor, Active Appearance Models, In H. Burkhardt and B. Neumann, editors, 5^{th} *European Conference on Computer Vision 1998* , Vol.2, pp. 484-498, Springer, Berlin, 1998. 13, 20

[2] T.F. Cootes, G.J. Edwards, C.J. Taylor, A Comparative Evaluation of Active Appearance Model Algorithms, In P. Lewis and M. Nixon, editors 9^{th} *British Machine Vision Conference*, Vol. 2, pp. 680-689, Southampton, UK, Sept. 1998. BMVA Press. 27, 31

[3] T.F. Cootes and P. Kittipanya-ngam, Comparing Variations on the Active Appearance Model Algorithm, *British Machine Vision Conference*, Cardiff University, pp. 837-846, Sept. 2002. 33

[4] X. Hou, S. Li, H. Zhang, Q. Cheng, Direct appearance models,*Computer Vision and Pattern Recognition Conference 2001*, volume 1, p. 828-833, 2001. 29

[5] H. van Kuilenburg, Expressions Exposed; Model Based Methods for Automatic Analysis of Face Images, *Masters thesis*, Department of Philosophy, Utrecht University, The Netherlands, 2005. 13, 19, 20

[6] T.F. Cootes and C.J Taylor. Statistical models of appearance for computer vision, *Technical report*, University of Manchester, Wolfson Image Analysis Unit, Imaging Science and Biomedical Engineering, March 2004. 13, 16, 21, 22

[7] A. Pentland and T. Choudhury, Face Recognition for Smart Environments, *IEEE Computer*, vol. 33, no. 2, pp. 50-55, Feb. 2000. 9

[8] S. Baker and I. Matthews, Equivalence and Efficiency of image alignment algorithms. *Computer Vision and Pattern Recognition Conference 2001*, vol. 1, pp. 1090-1097, 2001. 30