Automatic Construction of Active Appearance Models as an Image Coding Problem

Simon Baker, Iain Matthews, and Jeff Schneider

The Robotics Institute
Carnegie Mellon University
Pittsburgh, PA 15213

Abstract

The automatic construction of Active Appearance Models (AAMs) is usually posed as finding the location of the base mesh vertices in the input training images. In this paper, we re-pose the problem as an energy-minimizing image coding problem and propose an efficient gradient-descent algorithm to solve it.

Keywords: Active Appearance Models, automatic construction, unsupervised learning, image coding, inverse compositional image alignment, quadratic smoothness priors.

1 Introduction

Active Appearance Models (AAMs) [5], and the closely related concepts of Active Blobs [21] and 3D Morphable Models (3DMMs) [3, 15, 22], are non-linear, generative models of a certain visual phenomenon. The most frequent application of AAMs to date has been face modeling [17], however AAMs and 3DMMs may be useful for modeling other phenomena as well [15, 21].

AAMs are normally constructed from training images in which the location of the AAM shape mesh vertices are *hand-marked*. Recently, the task of automatically constructing AAMs (i.e. without hand-marking the mesh) has received a great deal of attention. The most common approach is to use some form of motion estimation to pre-compute the location of the model mesh in the training images. One such technique is optical flow [4, 15]. A very closely related approach is to non-rigidly align the training data and the model [20]. Finally, feature-point tracking has also been used [23]. All of these approaches are pragmatic, and lack elegance. Ultimately, they use an ad-hoc motion estimation technique to estimate the training data that would have been marked by hand. Hence, they are limited by the assumptions made by the motion estimation algorithms. The optical flow approaches only work when the optical flow algorithms work and the feature point tracking algorithms only work when there are strong, well-defined features to track.

In this paper we pose the problem of automatically constructing an AAM as an image coding problem. In this respect, the closest related work is that of Davies *et al.* who posed the problem of automatically constructing a statistical shape model (as opposed to an AAM) as a coding problem [6, 7]. Image coding is the task of representing a set of images as accurately as possible with a fixed number of parameters (or bits, even). Perhaps the most well-known image coding problem is the one that leads to Principal Components Analysis (PCA) [13]. Even though PCA is optimal (in a certain sense), its coding power is limited. Often a very large number of parameters are needed to code a set of images accurately. The problem with PCA is that the coding is linear in the pixel

intensities. Non-linear coding schemes, such as ones based on AAMs, can be far more powerful.

Another closely related body of work is the unsupervised learning literature where a number of approaches have recently been proposed that simultaneously solve for an appearance model and the parameters of a transformation relating the images [10, 11, 16]. Most of these approaches, however, assume that the set transformations, or shape component, is given *a priori*; i.e. it is independent of the training data rather than being learnt from the training data. When constructing an AAM, however, the shape basis is unknown and must be solved for. More recently there has been some work on simultaneously learning both a shape model and an appearance model [12, 14]. This work is perhaps the most closely related work to ours in the unsupervised learning literature. Our algorithm can be regarded as solving a similar problem to that of [12, 14], but for the more powerful AAM model of Cootes and Taylor [5], with all the benefi ts that entails. In a sense, it is a combination of the AAM modeling work of [5] and the unsupervised learning work of [14]. Our algorithm is also closely related to the unsupervised learning of generative, graphical models [8, 9].

2 Background: Linear Coding and PCA

Suppose we are given a set of N example training images from the set to be coded: $I^i(\mathbf{x})$ where $i = 1, 2, \ldots, N$ and where $\mathbf{x} = (x, y)^{\mathrm{T}}$ are the pixel coordinates. The coding goal is to represent these images as accurately as possible using the following *Linear Appearance Model*:

$$\mathrm{LAM}(\mathbf{x}; \boldsymbol{\lambda}) \equiv A_0(\mathbf{x}) + \sum_{j=1}^{n} \lambda_j A_j(\mathbf{x}) \tag{1}$$

where $A_0(\mathbf{x}), A_1(\mathbf{x}), \ldots, A_n(\mathbf{x})$ are a set of constant *basis images* and $\boldsymbol{\lambda} = (\lambda_1, \ldots, \lambda_n)^{\mathrm{T}}$ are a set of *coding parameters*. In common vision terminology "appearance" means pixel intensities $I^i(\mathbf{x})$ or $\mathrm{LAM}(\mathbf{x}; \boldsymbol{\lambda})$ [19]. Note that without loss of generality we can assume that the basis images $A_j(\mathbf{x})$ are orthonormal.

The *goal* of coding is to make the model $\mathrm{LAM}(\mathbf{x};\boldsymbol{\lambda})$ "as close as possible" to each image $I^i(\mathbf{x})$. We use the Euclidean L2 norm to measure "as close as possible" and so minimize:

$$\boldsymbol{\lambda^i} \equiv (\lambda_1^i,\ldots,\lambda_n^i)^{\mathrm{T}} \equiv \arg\min_{\boldsymbol{\lambda}} \sum_{\mathbf{x}\in I^i} \left[I^i(\mathbf{x}) - \mathrm{LAM}(\mathbf{x};\boldsymbol{\lambda})\right]^2 \tag{2}$$

where (with a slight abuse of terminology) the summation is performed over all of the pixels in the images I^i. (As is normal with PCA, we assume that all the images I^i are the same size.)

2.1 Coding and Decoding

Suppose for now that the basis images $A_j(\mathbf{x})$ are known. *Coding* an image $I^i(\mathbf{x})$ is then the process of estimating the coding parameters $\boldsymbol{\lambda^i} = (\lambda_1^i,\ldots,\lambda_n^i)^{\mathrm{T}}$ for that image; i.e. performing the minimization in Equation (2). The solution to this least squares problem is:

$$\lambda_j^i = \sum_{\mathbf{x}\in I^i} A_j(\mathbf{x}) \left[I^i(\mathbf{x}) - A_0(\mathbf{x})\right]. \tag{3}$$

Coding an image is little more than n image dot-products which can be performed very efficiently.

Decoding an image is reversing this process; i.e. generating the model instance $\mathrm{LAM}(\mathbf{x};\boldsymbol{\lambda})$ by evaluating Equation (1). Decoding an image is therefore also a very efficient process.

2.2 Optimal Coding: Principal Components Analysis

We now address choice of the basis images $A_j(\mathbf{x})$. This question is normally posed as one of minimizing the total coding error in Equation (2) across all of the example images $I^i(\mathbf{x})$:

$$\arg\min_{A_j(\mathbf{x})} \sum_{i=1}^{N} \left(\min_{\boldsymbol{\lambda}} \sum_{\mathbf{x}\in I^i} \left[I^i(\mathbf{x}) - \mathrm{LAM}(\mathbf{x};\boldsymbol{\lambda})\right]^2 \right). \tag{4}$$

It is well known that the minimum of this expression is attained when the basis images $A_j(\mathbf{x})$ are the Principal Components of the images $I^i(\mathbf{x})$ [13].

3 Coding with AAMs

The linear coding problem in Section 2 has the nice property that there is a closed-form solution for the optimal Linear Appearance Model (i.e. the basis functions.) The coding power of the optimal Linear Appearance Model can be very weak, however. We now consider coding with AAMs.

3.1 Active Appearance Models

There are two components to an *Active Appearance Model* (AAM) [5, 18], its shape and its appearance. We first define each component in turn and then how to generate a model instance. (We follow the notation of [18], and as in that paper, do not perform a joint shape/appearance PCA.)

3.1.1 Shape

The *shape* of an AAM is defined by the vertex locations of a triangulated mesh. The shape $\mathbf{s}$ of an AAM is a vector of the x and y-coordinates of the v vertices that make up the mesh:

$$\mathbf{s} \equiv (x_1, y_1, x_2, y_2, \ldots, x_v, y_v)^{\mathrm{T}}. \tag{5}$$

AAMs allow linear shape variation; i.e. the shape $\mathbf{s}$ can be expressed as a base shape $\mathbf{s}_0$ plus a linear combination of m shape vectors $\mathbf{s}_j$:

$$\mathbf{s} \equiv \mathbf{s}_0 + \sum_{j=1}^{m} p_j \mathbf{s}_j \tag{6}$$

where the coefficients p_j are the shape parameters. See Figure 1(d) for an example. As in Section 2, wherever necessary we assume that the shape vectors $\mathbf{s}_j$ are orthonormal.

3.1.2 Appearance

As a convenient abuse of terminology, let $\mathbf{s}_0$ also denote the pixels $\mathbf{x} = (x, y)^{\mathrm{T}}$ that lie inside the base mesh $\mathbf{s}_0$. The *appearance* of an AAM is an image $A(\mathbf{x})$ defined over the pixels in the base

mesh $\mathbf{x} \in \mathbf{s}_0$. AAMs allow linear appearance variation; i.e. the appearance $A(\mathbf{x})$ can be expressed as a base appearance $A_0(\mathbf{x})$ plus a linear combination of n appearance images $A_j(\mathbf{x})$:

$$A(\mathbf{x}) \equiv A_0(\mathbf{x}) + \sum_{j=1}^{n} \lambda_j A_j(\mathbf{x}) \qquad \forall \mathbf{x} \in \mathbf{s}_0 \tag{7}$$

where the coefficients λ_j are the appearance parameters. See Figure 1(c) for an example. As in Section 2, wherever necessary we assume that the images A_j are orthonormal.

3.1.3 Generating a Model Instance: Decoding

Given the shape parameters $\mathbf{p} = (p_1, p_2, \ldots, p_m)^{\mathrm{T}}$, Equation (6) can be used to compute the shape $\mathbf{s}$. Similarly, the appearance parameters $\boldsymbol{\lambda} = (\lambda_1, \lambda_2, \ldots, \lambda_n)^{\mathrm{T}}$ can be used to compute the appearance $A(\mathbf{x})$. The AAM model instance is then computed by warping the appearance $A(\mathbf{x})$ from the base mesh $\mathbf{s}_0$ onto the model shape $\mathbf{s}$. In particular, the pair of meshes $\mathbf{s}_0$ and $\mathbf{s}$ define a piecewise affine warp from $\mathbf{s}_0$ to $\mathbf{s}$ which we denote $\mathbf{W}(\mathbf{x}; \mathbf{p})$. The AAM model instance is computed by *backwards warping* the appearance A from $\mathbf{s}_0$ to $\mathbf{s}$ as follows:

$$\mathrm{AAM}(\mathbf{x}; \mathbf{p}; \boldsymbol{\lambda}) \equiv A(\mathbf{W}^{-1}(\mathbf{x}; \mathbf{p})) = A_0(\mathbf{W}^{-1}(\mathbf{x}; \mathbf{p})) + \sum_{j=1}^{n} \lambda_j A_j(\mathbf{W}^{-1}(\mathbf{x}; \mathbf{p})) \ \forall \mathbf{x} \in \mathbf{s}. \tag{8}$$

Given a pixel $\mathbf{x}$ in $\mathbf{s}$, the origin of this pixel under the warp is the pixel $\mathbf{W}^{-1}(\mathbf{x}; \mathbf{p})$ in $\mathbf{s}_0$. The appearance model is sampled at this point and $\mathrm{AAM}(\mathbf{x}; \mathbf{p}; \boldsymbol{\lambda})$ set to that value.

3.2 AAM Fitting: Image Coding

Analogously to Section 2.1, the goal of coding an image $I^i(\mathbf{x})$ with an AAM is to minimize the Euclidean L2 error between the image and the model:

$$\arg\min_{\mathbf{p}, \boldsymbol{\lambda}} \sum_{\mathbf{x} \in I^i} \left[I^i(\mathbf{x}) - \mathrm{AAM}(\mathbf{x}; \mathbf{p}; \boldsymbol{\lambda}) \right]^2 \tag{9}$$

i.e. fit the AAM to I^i. Because AAMs are non-linear in their shape parameters $\mathbf{p}$, Equation (9) is a non-linear optimization problem. Coding an image is therefore subject to all of the difficulties associated with non-linear optimization, primarily local minima.

Another issue is computational efficiency. Performing a non-linear optimization can be a slow process, especially with image sized data. Fortunately we have recently proposed an efficient algorithm for fitting (coding with) AAMs [18]. Our algorithm actually minimizes:

$$\arg\min_{\mathbf{p},\boldsymbol{\lambda}} \sum_{\mathbf{x}\in\mathbf{s}_0} \left[I^i(\mathbf{W}(\mathbf{x};\mathbf{p})) - \left(A_0(\mathbf{x}) + \sum_{j=1}^{n} \lambda_j A_j(\mathbf{x}) \right) \right]^2 . \tag{10}$$

There are two differences between Equations (9) and (10): (1) the warp $\mathbf{W}(\mathbf{x};\mathbf{p})$ estimated in one is the inverse of that estimated in the other, and (2) the error is computed in different reference frames, the input image I^i and the model frame $\mathbf{s}_0$. The first of these differences is not important; the warp can be inverted after it has been estimated. The second difference is theoretically important; strictly the two error criteria weight the pixels differently and so the optimal solution will be slightly different. In practice, however, using Equation (10) and then inverting $\mathbf{W}(\mathbf{x};\mathbf{p})$ gives a good approximate solution to Equation (9). The benefit of coding an image this way is that the solution of Equation (10) can be performed far more efficiently. It can be performed in real time ($\approx$ 7ms or 150Hz) for typical AAMs [18]. Using Equation (10) is therefore approximate, but very efficient. If efficiency is not a concern, the straight-forward Gauss-Newton solution of Equation (9) can be used instead. The remainder of this paper is based on Equation (10) rather than Equation (9). A similar, but slower, coding algorithm can also be derived from Equation (9).

3.3 Optimal Coding/Automatic Construction of AAMs

Analogously to Section 2.2, we now ask what is the best choice for the AAM shape vectors $\mathbf{s}_j$, $j = 0, 1, \ldots, m$ and the AAM appearance images $A_j(\mathbf{x})$, $j = 0, 1, \ldots, n$. We pose this question

as minimizing the total coding error across all of the example images $I^i(\mathbf{x})$:

$$\arg \min_{\mathbf{s}_j, A_j(\mathbf{x})} \sum_{i=1}^{N} \left(\min_{\mathbf{p}, \boldsymbol{\lambda}} \sum_{\mathbf{x} \in \mathbf{s}_0} \left[I^i(\mathbf{W}(\mathbf{x}; \mathbf{p})) - \left(A_0(\mathbf{x}) + \sum_{j=1}^{n} \lambda_j A_j(\mathbf{x}) \right) \right]^2 \right). \tag{11}$$

Pulling the inner "min" outside the summation requires that we introduce an index on $\mathbf{p}$ and $\boldsymbol{\lambda}$ since a different set of coding parameters will in general be needed for for each example image $I^i(\mathbf{x})$. This leaves the optimal AAM coding problem as minimizing:

$$\arg \min_{\mathbf{s}_j, A_j(\mathbf{x}), \mathbf{p}^i, \boldsymbol{\lambda}^i} \sum_{i=1}^{N} \sum_{\mathbf{x} \in \mathbf{s}_0} \left[I^i(\mathbf{W}(\mathbf{x}; \mathbf{p}^i)) - \left(A_0(\mathbf{x}) + \sum_{j=1}^{n} \lambda_j^i A_j(\mathbf{x}) \right) \right]^2. \tag{12}$$

Equation (12) is another way to pose the automatic AAM construction problem; i.e. find the AAM shape and appearance basis vectors that represent the training images as accurately as possible. Unlike the approach of trying to compute the correspondence between the training images and the model [4, 15, 20, 23], this formulation of the problem has a well defined optimality criterion. This approach is similar to that of Davies *et al.* [6, 7] who optimize a minimum description length criterion for statistical shape models. The algorithm of [6, 7] is of course not applicable to AAMs.

3.4 Solving the Optimal AAM Coding Problem

The minimization in Equation (12) is a huge non-linear optimization over a very large number of variables. Solving it requires far more effort than computing the eigenvectors of the covariance matrix (as in PCA). We solve it by iteratively computing A_j, $\boldsymbol{\lambda}^i$, $\mathbf{s}_j$, and $\mathbf{p}^i$ in turn, assuming where necessary that initial estimates of the others are available. We initialize the algorithm by setting $\mathbf{s}_j$ and $\mathbf{p}^i$ to be zero and using PCA to estimate A_j and $\boldsymbol{\lambda}^i$. We also need to specify the number of shape components m. Like PCA, the algorithm simultaneous generates results for any desired range of n, the number of appearance parameters. Since the algorithm is quite efficient, it can

be run multiple times for different settings of m to obtain the best trade off between shape and appearance. The components of the AAM (A_j, $\boldsymbol{\lambda}^i$, $\mathbf{s}_j$, and $\mathbf{p}^i$) are then updated in turn as follows:

Updating A_j: If we know $\mathbf{s}_j$ and $\mathbf{p}^i$ we can compute the warp $\mathbf{W}(\mathbf{x};\mathbf{p}^i)$ between the base mesh $\mathbf{s}_0$ and the mesh $\mathbf{s} = \mathbf{s}^i$ for each input image I^i. The problem then reduces to a warped version of the original linear coding problem. (When $\mathbf{p}^i = \mathbf{0}$ this is the original optimal linear coding problem.) We warp each image onto the base mesh to give $I^i(\mathbf{W}(\mathbf{x};\mathbf{p}^i))$. We then perform PCA on these vectors, setting $A_0(\mathbf{x})$ to be the mean vector and $A_j(\mathbf{x})$, $j = 1,\ldots,n$, to be the eigenvectors of the covariance matrix with the n largest eigenvalues.

Updating $\boldsymbol{\lambda}^i$: If $\mathbf{s}_j$ and $\mathbf{p}^i$ are known, we can again compute $\mathbf{W}(\mathbf{x};\mathbf{p}^i)$. If A_j are also known, we can then compute $\boldsymbol{\lambda}^i$ with the warped equivalent of Equation (3):

$$\lambda_j^i \;=\; \sum_{\mathbf{x}\in\mathbf{s}_0} A_j(\mathbf{x}) \left[I^i(\mathbf{W}(\mathbf{x};\mathbf{p}^i)) - A_0(\mathbf{x}) \right]. \tag{13}$$

Updating $\mathbf{s}_j$: We first assume that the mesh shape $\mathbf{s}$ is completely free for every image I^i. Let $\mathbf{W}(\mathbf{x};\mathbf{s})$ denote the piecewise affine warp from the base mesh $\mathbf{s}_0$ to the mesh $\mathbf{s}$. We then compute a mesh $\mathbf{s}^i$ for each image I^i by minimizing:

$$\mathbf{s}^i \;=\; \arg\min_{\mathbf{s}} \sum_{\mathbf{x}\in\mathbf{s}_0} \left[I^i(\mathbf{W}(\mathbf{x};\mathbf{s})) - \left(A_0(\mathbf{x}) + \sum_{j=1}^{n} \lambda_j^i A_j(\mathbf{x}) \right) \right]^2 \tag{14}$$

using our AAM fitting algorithm [18]. We then compute $\mathbf{s}_j$, $j = 0,\ldots,m$ by performing PCA on the vectors $\mathbf{s}^i$, setting $\mathbf{s}_0$ to be the mean vector and $\mathbf{s}_j$ to the eigenvectors of the covariance matrix with the m largest eigenvalues.

Updating $\mathbf{p}^i$: If A_j and $\mathbf{s}_j$ are known, this task is just a special instance of the AAM fitting algorithm. We use the algorithm in [18] to compute:

$$\mathbf{p}^i \;=\; \arg\min_{\mathbf{p}} \sum_{\mathbf{x}\in\mathbf{s}_0} \left[I^i(\mathbf{W}(\mathbf{x};\mathbf{p})) - \left(A_0(\mathbf{x}) + \sum_{j=1}^{n} \lambda_j^i A_j(\mathbf{x}) \right) \right]^2. \tag{15}$$

The algorithm we have just described is a non-linear optimization and just like any other non-linear optimization is prone to falling into local minima. There are a variety of techniques that can be used to help avoid local minima in image alignment tasks such as those in Equations (14) and (15). Typical examples include processing on a Gaussian pyramid and using progressive transformation complexity [2]. As well as using these heuristics, we add one more to Equation (14). Instead of optimizing Equation (14) we actually optimize

$$\mathbf{s}^i = \arg\min_{\mathbf{s}} \sum_{\mathbf{x} \in \mathbf{s}_0} \left[I^i(\mathbf{W}(\mathbf{x};\mathbf{s})) - \left(A_0(\mathbf{x}) + \sum_{j=1}^{n} \lambda_j^i A_j(\mathbf{x}) \right) \right]^2 + \mathbf{s}^{\mathrm{T}} Q \mathbf{s} \tag{16}$$

where $\mathbf{s}^{\mathrm{T}} Q \mathbf{s}$ is a quadratic form which encourages the mesh $\mathbf{s}$ to deform smoothly. (We also project out the component of Q in the subspace corresponding to the previous estimate of the shape vectors $\mathbf{s}_j$ to allow the mesh to move freely in that space.) Note that to minimize the quantity in Equation (16) our AAM fitting algorithm [18] has to be modified slightly [1].

3.5 Experimental Results

In Figure 1 we present the results of running our algorithm on a set of 100 randomly generated square patterns. Each pattern consists of 4 equally sized squares arranged to produce a larger square. The squares have randomly generated translations, rotations, scales, and intensities. Figure 1(a) includes 4 example inputs. Because this data mostly consists of constant intensity regions, using optical flow to align the inputs would not be possible. Hence, methods such as [4, 15] are not applicable to this data. Also, since the input is a random collection of images, rather than a video sequence, using feature point tracking techniques like [23] would also not be possible. Finally, it would even be hard to manually ground-truth this data. We use a mesh consisting of 25 vertices. See Figure 1(d). The "correct" location of every one of these "landmarks" in every training image is in the middle of a constant intensity region. It is therefore essentially impossible to mark by

(a) Example Inputs (4 of 100)

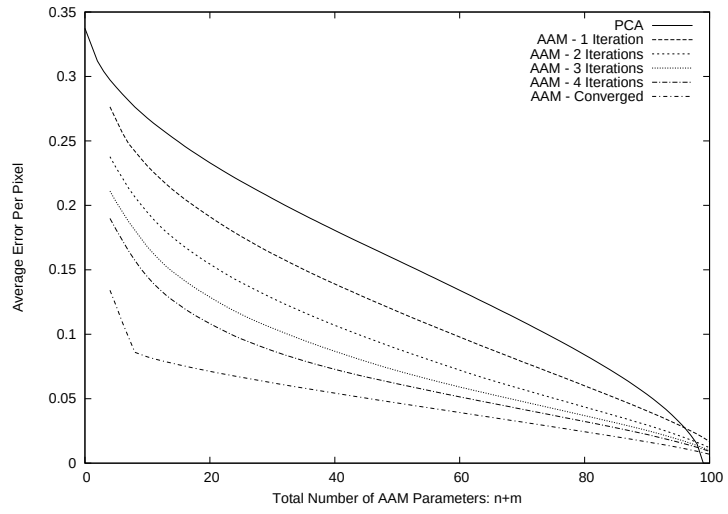
(b) PCA Results: A_0-A_3

(c) AAM Appearance: A_0-A_3

(d) AAM Shape $\mathbf{s}_0$, $\mathbf{s}_0 + 50\mathbf{s}_1$ $\mathbf{s}_0 + 50\mathbf{s}_2$, $\mathbf{s}_0 + 50\mathbf{s}_3$

(e) PCA Reconstructed (8 Appearance Parameters)

(f) AAM Reconstructed (8 Parameters, 4 Shape, 4 Appearance)



(g) Error Plot

Figure 1: Experimental results for a set of randomly generated square patterns. See text for details.

hand. In fact, this is one of the additional benefits of our automatic algorithm: it does not require that the mesh vertices correspond to "semantically meaningful" points in the images.

The results of performing PCA on the data in Figure 1(a) are included in Figure 1(b) and the results of reconstructing the input images with 8 eigenvectors are shown in Figure 1(e). The AAM computed with our algorithm is shown in Figures 1(c) and (d). Figure 1(c) includes the appearance variation A_0, A_1, A_2, A_3. Figure 1(d) illustrates the shape variation. Specifically we plot: $\mathbf{s}_0$, $\mathbf{s}_0 + 50 \times \mathbf{s}_1$, $\mathbf{s}_0 + 50 \times \mathbf{s}_2$, and $\mathbf{s}_0 + 50 \times \mathbf{s}_3$. The results of reconstructing the input images using the AAM with 8 parameters (4 shape, 4 appearance) are shown in Figure 1(f). Finally, Figure 1(g) plots the RMS coding error (the square root of either Equation (4) or (12)) per pixel as a function of the number of combined AAM shape and appearance parameters. (Note that the dynamic range of each parameter is different and so a different number of bits would be needed to code each parameter, with a given accuracy. Our application is model building, however, not image coding, and we only aim to minimize the number of parameters, not the total number of bits.)

Studying Figure 1(g), and comparing Figures 1(e) and (f), we see that the automatically constructed AAM represents the input images very well. Also, studying Figures 1(c) and (d) we see that our algorithm has "learnt" that the input images consist of 4 equally sized squares with different randomly generated intensities (see Figure 1(c)) that can be translated, rotated, and scaled (see Figure 1(d)). This solution is intuitively the optimal solution. The only reason that the coding error in Figure 1(g) does not become exactly zero after $n + m = 8$ parameters is because of the interpolation errors around the edges of the squares than can be best seen in Figure 1(c). In the accompanying movie "ws.mpg" we include an illustration of the algorithm running on this data. In the movie, the top-left panel displays the input image overlayed with the current estimate of the AAM mesh for that image. The remainder of the top row displays the current estimate of the shape in the same format as Figure 1(d). The bottom row displays the current estimate of the AAM

appearance variation in the same format as Figure 1(c). For the data in Figure 1 our algorithm converged in 10 iterations, each iteration taking approximately 45 seconds on a 2.0 GHz P4.

In Figure 2 we present the results of our algorithm on 200 images of a face. Figure 2(b) includes the mean appearance and the first 4 PCA eigenvectors. Figure 2(c) includes the first AAM appearance eigenvector and the first 3 shape eigenvectors. Again, the AAM coding (see Figure 2(e)) is far more accurate than the PCA coding (see Figure 2(d) and (f))[1]. The accompanying movie "face.mpg" includes an illustration of the algorithm running on this face data.

In Figure 2(e) we also overlay the locations of the two eyes and the mouth computed by warping their locations from the mean face using the piecewise affine warp for each input image. These results illustrate how our algorithm implicitly computes the correspondence between the input images. Finally, we computed a measure of the generalizability of the automatically computed AAM. We computed how well the AAM could explain a subset of 4 images withheld from the training data (but from the same sequence.) The RMS pixel error for the 4 images was 3.8 grey-levels, a substantial improvement over the same measure for PCA of 9.5 grey-levels.

4 Conclusion

We have posed the automatic construction of AAMs as an image coding problem. This approach should be compared with the traditional approach of attempting to compute the correspondence between the training samples and the AAM mesh and then computing the AAM in the normal manner [4, 15, 20, 23]. Our approach is more elegant, being posed in terms of a well-defined optimality criterion (see Equation (12)), and more general, since it can operate on data (such as Figure 1) for which neither optical flow nor feature point tracking is possible. On the other hand, our formulation does lead to a huge non-linear optimization. Although there are a variety of heuristics

[1] As above, note that in Figure 2(d) a coding error of zero is not possible (without a huge number of appearance vectors) due to bilinear interpolation error. A coding error of just under 0.1 is probably the lowest achievable.

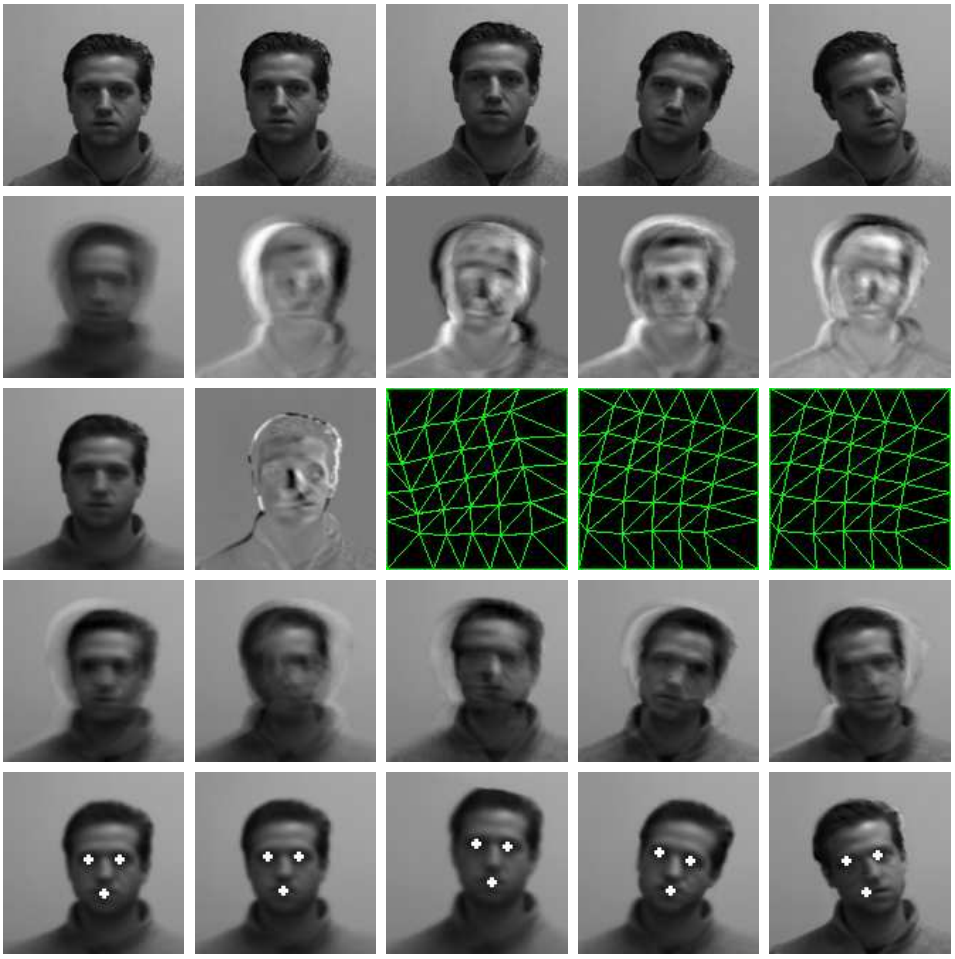
(a) Inputs (5 of 200)

(b) PCA Results

(c) AAM

(d) PCA Recons.

(e) AAM Recons.



(f) Error Plot

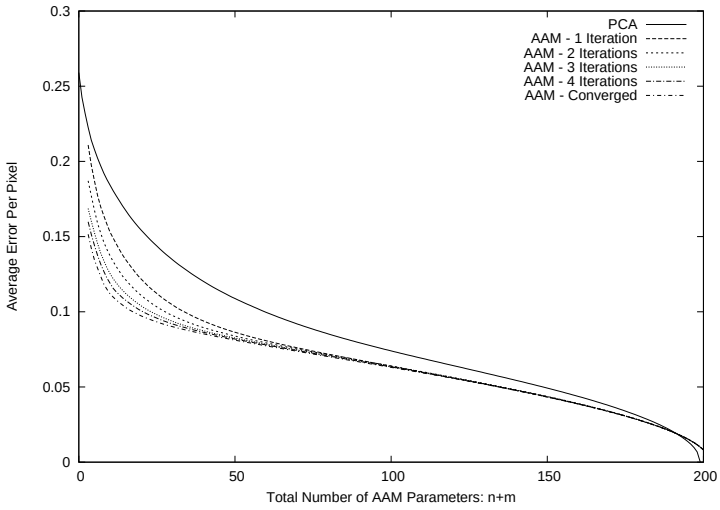


Figure 2: Results on a set of face images. The input consists of 200 face images like the 5 examples in (a). The output (c) consists of 3 AAM shape vectors and 1 AAM appearance image. Comparing the AAM reconstruction using 3 shape and 1 appearance parameters (e) against a PCA reconstruction with 4 parameters (d), we find that the automatically computed AAM coding is far more accurate than the optimal linear coding (PCA). The quantitative coding error is shown in (f) and the PCA eigenvectors in (b).

to help avoid local minima (see Section 3.4), avoiding them is still a difficult task, and ultimately for some datasets the more pragmatic approach may be the correspondence based approach.

Acknowledgments

The research described in this paper was supported by the U.S. Department of Defense through award N41756-03-C4024. We would also like to thank the reviewers of the (rejected) NIPS 2003 version of this paper for their comments. We believe that their feedback has greatly improved the paper. We also thank Emo Todorov for his encouragement to publish this work.

References

[1] S. Baker, R. Gross, and I. Matthews. Lucas-Kanade 20 years on: A unifying framework: Part 4. Technical Report CMU-RI-TR-04-14, CMU Robotics Institute, 2004.

[2] J. Bergen, P. Anandan, K. Hanna, and R. Hingorani. Hierarchical model-based motion estimation. In *Proc. of the European Conference on Computer Vision*, pages 237–252, 1992.

[3] V. Blanz and T. Vetter. A morphable model for the synthesis of 3D faces. In *Computer Graphics, Annual Conference Series (SIGGRAPH)*, pages 187–194, 1999.

[4] M. Brand. Morphable 3D models from video. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, volume 2, pages 456–463, 2001.

[5] T. Cootes, G. Edwards, and C. Taylor. Active appearance models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(6):681–685, 2001.

[6] R. Davies, C. Twining, P. Allen, T. Cootes, and C. Taylor. Shape discrimination in the hippocampus using an MDL model. In *Proceedings of the International Conference on Information Processing in Medical Imaging*, pages 38–50, 2003.

[7] R. Davies, C. Twining, T. Cootes, and C. Taylor. A minimum description length approach to statistical shape modelling. *IEEE Transactions on Medical Imaging*, 21:525–537, 2002.

[8] P. Dayan, G. Hinton, R. Neal, and R. Zemel. The Helmholtz machine. *Neural Computation*, 7:1022–1037, 1995.

[9] B. Frey and G. Hinton. Efficient stochastic source coding and an application to a Bayesian network source model. *The Computer Journal*, 40:157–165, 1997.

[10] B. Frey and N. Jojic. Esimtating mixture models of images and inferring spatial transforms using the em algorithm. In *Proceeding of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1416–1422, 1999.

[11] B. Frey and N. Jojic. Transformed component analysis: Joint estimation of spatial transformations and image components. In *Proceedings of the IEEE International Conference on Computer Vision*, volume 2, pages 1190–1196, 1999.

[12] B. Frey, A. Kanna, and N. Jojic. Product Analysis: Learning to model observations as products of hidden variables. In *Neural Information Processing Systems*, pages 729–735, 2001.

[13] K. Fukunaga. *Introduction to statistical pattern recognition*. Academic Press, 1990.

[14] N. Jojic, P. Simard, B. Frey, and D. Heckerman. Separating appearance from deformation. In *Proceedings of the IEEE International Conference on Computer Vision*, volume 2, pages 288–294, 2001.

[15] M. Jones and T. Poggio. Multidimensional morphable models: A framework for representing and matching object classes. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 683–688, 1998.

[16] F. De la Torre. Automatic learning of appearance face models. In *Proceedings of the Workshop on Recognition, Analysis and Tracking of Faces and Gestures in Realtime Systems*, pages 32–39, 2001.

[17] A. Lanitis, C. J. Taylor, and T. F. Cootes. Automatic interpretation and coding of face images using flexible models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(7):742–756, 1997.

[18] I. Matthews and S. Baker. Active Appearance Models revisited. *International Journal of Computer Vision*, 60(2), 2004. In Press.

[19] H. Murase and S.K. Nayar. Visual learning and recognition of 3-D objects from appearance. *International Journal of Computer Vision*, 14:5–24, 1995.

[20] D. Rueckert, A. Frangi, and J. Schnabel. Automatic construction of 3D statistical deformation models using non-rigid registration. In *Proceedings of Medical Image Computing and Computer-Assisted Intervention*, pages 77–84, 2001.

[21] S. Sclaroff and J. Isidoro. Active blobs. In *Proceedings of the 6th IEEE International Conference on Computer Vision*, pages 1146–1153, 1998.

[22] T. Vetter and T. Poggio. Linear object classes and image synthesis from a single example image. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 19(7):733–741, 1997.

[23] K. Walker, T. Cootes, and C. Taylor. Automatically building appearance models from image sequences using salient features. *Image and Vision Computing*, 20(5–6):435–440, 2002.