Image Coding with Active Appearance Models

Simon Baker, Iain Matthews, and Jeff Schneider

CMU-RI-TR-03-13

The Robotics Institute
Carnegie Mellon University

Abstract

Image coding is the task of representing a set of images as accurately as possible using a fixed number of parameters. One well known example is the linear coding problem that leads to Principal Components Analysis (PCA). Although optimal in a certain sense, PCA has limited coding power. A large number of parameters are often required to code a set of images accurately. In this paper we consider image coding using Active Appearance Models (AAMs). AAMs are a class of generative non-linear models (although linear in both shape and appearance) which have received a great deal of recent attention in the computer vision literature. We describe what it means to code and decode with AAMs, pose the optimal coding problem, and propose an algorithm to solve it. Our algorithm can also be interpreted as an unsupervised model building algorithm.

Keywords: Active Appearance Models, Automatic Construction, Image Coding.

1 Introduction

Image coding is the task of representing a set of images as accurately as possible using a fixed number of parameters. Perhaps the most well-known image coding problem is the one that leads to Principal Components Analysis (PCA) [6]. With PCA each image is coded as a linear combination of a fixed number of constant basis images. Although this coding scheme is simple, it has two important properties. First, the coding problem has a closed-form solution. The optimal basis images are the eigenvectors with the largest eigenvalues of the covariance matrix of the set of images. Second, image coding (estimating the coding parameters for an image) and image decoding (regenerating the image from the parameters) are simple linear operations and so can be performed very efficiently.

On the other hand, 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 image appearance. (In the vision community, the term "appearance" is generally used to mean "pixel intensities" [8].) If we consider non-linear schemes we may be able to find a far more powerful scheme.

The set of all non-linear coding schemes is far too general to consider. Finding the optimal scheme would be impossible. Even representing all of the possibilities would be very challenging. In this paper we consider coding schemes based on Active Appearance Models (AAMs) [4]. AAMs are an instance of a "linear shape-plus-appearance model." The parameters of an AAM can be divided into a set of shape parameters and a set of appearance parameters. The shape part of the model is linear in terms of the shape parameters. The appearance part of the model is linear in terms of the appearance parameters. When the shape and appearance are combined, the final model pixels are non-linear in terms of the shape parameters (although still linear in the appearance parameters.)

In this paper we investigate image coding using AAMs. We describe what it means to code and decode an image using this scheme. We also pose the optimal coding problem, analogously to the problem solved by PCA, and propose an algorithm to solve it. We begin with a brief review of linear coding and Principal Components Analysis (PCA).

2 Background: Coding with Linear Appearance Models: PCA

Suppose we are given a set of N example 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. Our 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 $I^{\boldsymbol{i}}(\mathbf{x})$ or $\mathrm{LAM}(\mathbf{x};\boldsymbol{\lambda})$ [8]. In Equation (1) these intensities are linear functions of the coding parameters λ_j which is why we refer to it as a Linear Appearance Model. Note that without loss of generality we can assume that the basis images $A_j(\mathbf{x})$ are orthonormal. If they are not orthonormal, we orthonormalize them using Gramm-Schmidt. The resulting model is equivalent in the sense that $\mathrm{LAM}(\mathbf{x};\boldsymbol{\lambda})$ spans the same subspace of images.

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

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

where (with a slight abuse of terminology) the summation is performed over all of the pixels in the images $I^{\boldsymbol{i}}$. (As in PCA, we assume that all $I^{\boldsymbol{i}}$ have the same size and shape.)

2.1 Coding and Decoding

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

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

(assuming that the basis images $A_j(\mathbf{x})$ are orthonormal.) Coding an image is therefore 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 Linear Coding: Principal Components Analysis

We now address the question of what is the best choice for 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 $I^i(\mathbf{x})$; i.e. to minimize:

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

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 $I^i(\mathbf{x})$; i.e. $A_0(\mathbf{x})$ is the mean of the N example images $I^i(\mathbf{x})$ and the other basis images $A_j(\mathbf{x})$, $j = 1, \ldots n$, are the eigenvectors with the n largest eigenvalues of the covariance matrix of the example images $I^i(\mathbf{x})$ [6].

3 Coding with Linear Shape-Plus-Appearance Models: 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; i.e. for typical values of n the coding error in Equation (4) can be quite large. (See Figure 1(e).) We now consider a non-linear coding strategy that is far more appropriate for typical image data.

3.1 Linear Shape-Plus-Appearance Models

There are two components to a *linear shape-plus-appearance model* (often known in the vision community as an *Active Appearance Model* (AAM) [4, 7]), its shape and its appearance. We first define each component in turn and then how to generate a model instance.

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. As for the linear appearance models in Section 2, this means that 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 $\mathbf{W}(\mathbf{x}; \mathbf{p})$. For each triangle in $\mathbf{s}_0$ there is a corresponding triangle in $\mathbf{s}$. Each pair of triangles defines a unique affine warp such that the vertices of the first triangle map to the vertices of the second. The AAM model instance is then computed by *backwards warping* the appearance A from $\mathbf{s}_0$ to $\mathbf{s}$. This process is defined by the following equation:

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

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:

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

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 [7]. Our algorithm actually minimizes:

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

The differences between Equations (9) and (10) are twofold: (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 summed over different images, I^i versus $\mathbf{s}_0$. 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 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 [7]. Using Equation (10) to code images is therefore approximate, but very efficient. If efficiency is not a concern, the straight-forward Gauss-Newton or Levenberg-Marquardt solution of Equation (9) can be used instead.

3.3 Optimal Coding with AAMs

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 $A_j(\mathbf{x}), j = 0, 1, \ldots, n$. As in Section 2.2, 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 be needed for for each example image $I^i(\mathbf{x})$ (in general). This leaves the optimal coding problem as minimizing:

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

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 substantially 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 is possible to run the algorithm 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. 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. (Note that when $\mathbf{p}^i = \mathbf{0}$ this reduces to the original optimal linear coding problem.)

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 [7]. 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 algo-

rithm. We use the algorithm in [7] 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 [7] 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. The results of performing PCA on this data 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 coding parameters. Studying Figure 1(g), and comparing Figures 1(e) and (f), we see that the AAM is far better able to code the input images. 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.

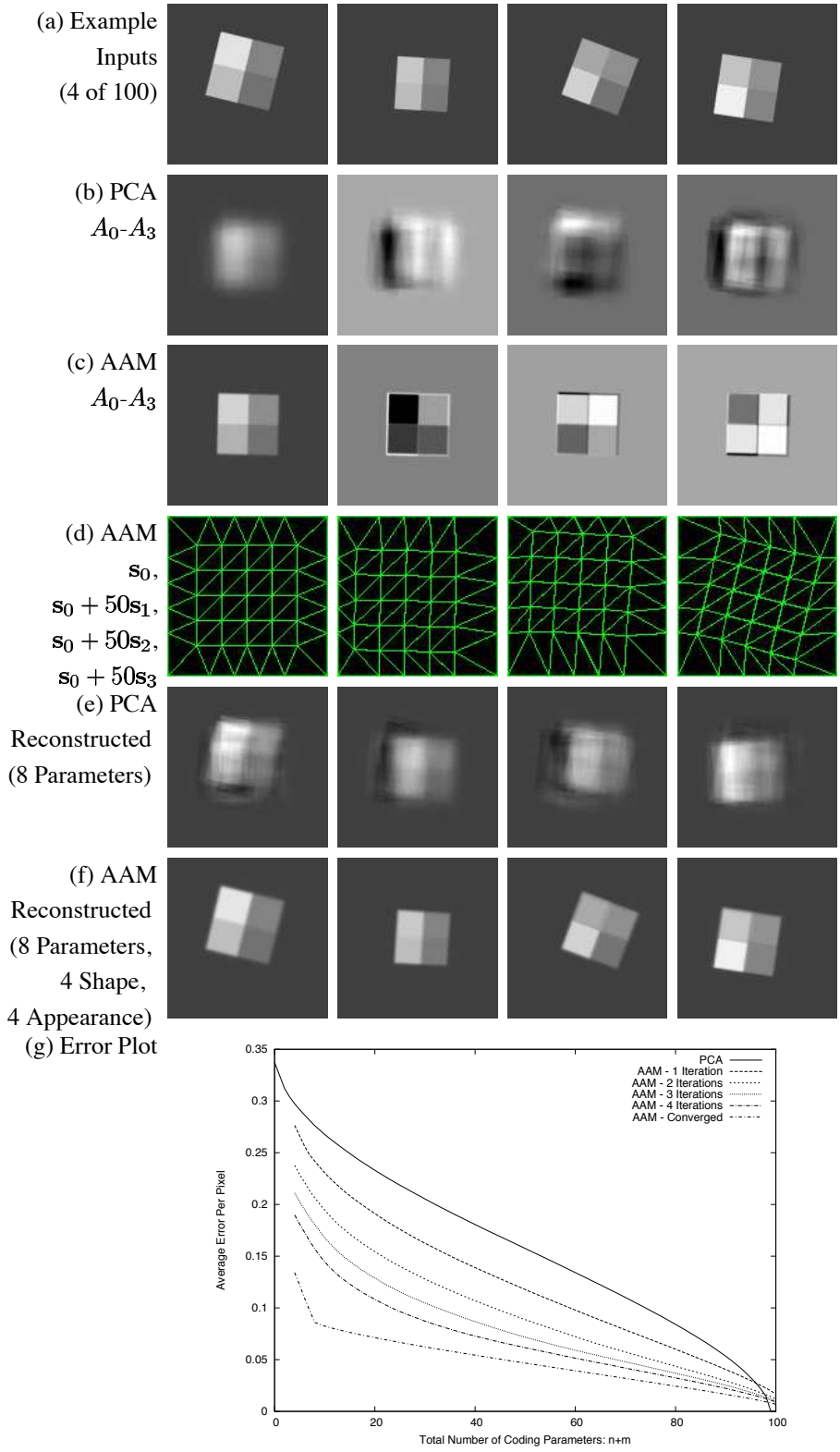

Figure 1: Experimental results for a set of randomly generated square patterns. See text for details.

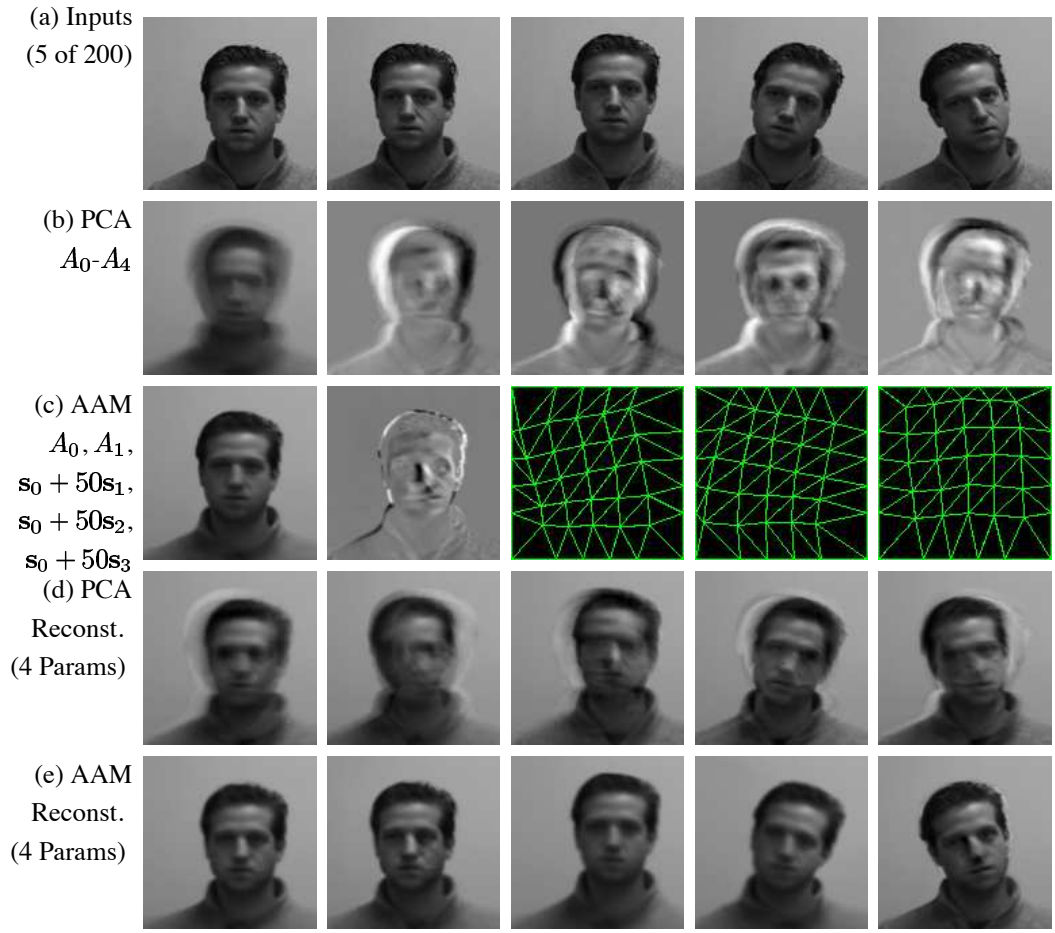

Figure 2: The results of running our algorithm on 200 images of a face. The input consists of 200 face images like the 5 examples in (a). The output of our algorithm (c) consists of the AAM shape vectors $\mathbf{s}_j$, the AAM appearance images A_j, (and the coding parameters $\boldsymbol{\lambda}^i$ and $\mathbf{p}^i$ of the 200 images.) We compare the AAM reconstruction using 3 shape and 1 appearance parameters (e) against a PCA reconstruction with 4 parameters (d). The PCA eigenvectors are shown in (b). As can be seen, the automatically computed AAM coding is far more accurate than the optimal linear coding (PCA). The quantitative results are similar to those in Figure 1(g), but are omitted for lack of space. In particular, the average error per pixel for $n + m = 4$ parameters is reduced from 0.21 to 0.13.

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 other experiments (results omitted for lack of space), we have provided our algorithm with similar inputs, such as translated white squares. In each case, our algorithm "learns" to code the data with the optimal solution. For example, the translated white squares are coded as translated (2 shape parameters) white squares with varying intensity (1 appearance parameter.) For the data in Figure 1 our algorithm converged in 10 iterations, each iteration taking approximately 45 seconds on a 2.0 GHz P4. The total run time was about 7 minutes 30 seconds.

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 optimal linear coding (see Figure 2(d)). Note that the motion of the face in Figure 2(a) is mostly 2D. AAMs are a 2D model and are unable to model the occlusions occurring in 3D. We are currently extending AAMs to include occlusion reasoning which will hopefully allow us to extend our algorithm to 3D.

4 Conclusion and Related Work

We have generalized the linear (appearance) coding problem to the linear shape-plus-appearance (AAM) coding problem and have proposed a gradient descent algorithm to solve the optimal coding problem. Like all gradient descent algorithms, our algorithm is prone to local minima, although we have proposed the warp smoothness prior in Equation (16) to partially alleviate the problem. Our results show that, not only is linear shape-plus-appearance coding often far more accurate than linear coding, but the coding parameters are often far more semantically meaningful. Compare Figures 1(b), (c), and (d).

Our image coding algorithm can also be regarded as an unsupervised (AAM) model building algorithm. The task of unsupervised model building has recently received a great deal of interest. Although we do not have space for a comprehensive literature review, note that previous approaches are more restrictive, for example, assuming a known shape basis $\mathbf{s}_j$ [5], requiring the ability to detect anatomical features [9], or using 3D vision [3].

Acknowledgements

The research described in this report was by the U.S. Department of Defense through award N41756-03-C4024.

References

[1] S. Baker, R. Gross, and I. Matthews. Lucas-Kanade 20 years on: A unifying framework: Part 4. Technical Report (in preparation), CMU Robotics Institute, 2003.

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

[3] M. Brand. Morphable 3D models from video. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2001.

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

[5] B. Frey and N. Jojic. Esimtating mixture models of images and inferring spatial transforms using the em algorithm. In *Proc. of CVPR*, 1999.

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

[7] I. Matthews and S. Baker. Active appearance models revisited. Technical Report CMU-RI-TR-03-02, Carnegie Mellon University Robotics Institute, April 2003.

[8] 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.

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