Active Appearance Models Revisited

Iain Matthews and Simon Baker

CMU-RI-TR-03-02

The Robotics Institute
Carnegie Mellon University

Abstract

Active Appearance Models (AAMs) and the closely related concepts of Morphable Models and Active Blobs are generative models of a certain visual phenomenon. Although linear in both shape and appearance, overall, AAMs are nonlinear parametric models in terms of the pixel intensities. Fitting an AAM to an image consists of minimizing the error between the input image and the closest model instance; i.e. solving a nonlinear optimization problem. We propose an efficient fitting algorithm for AAMs based on the *inverse compositional* image alignment algorithm. We show how the appearance variation can be "projected out" using this algorithm and how the algorithm can be extended to include a "shape normalizing" warp, typically a 2D similarity transformation. We evaluate our algorithm to determine which of its novel aspects improve AAM fitting performance.

Keywords: Active Appearance Models, AAMs, Active Blobs, Morphable Models, fitting, efficiency, Gauss-Newton gradient descent, inverse compositional image alignment.

1 Introduction

Active Appearance Models (AAMs) [Cootes *et al.*, 2001], first proposed in [Cootes *et al.*, 1998], and the closely related concepts of Active Blobs [Sclaroff and Isidoro, 1998] and Morphable Models [Vetter and Poggio, 1997, Jones and Poggio, 1998, Blanz and Vetter, 1999], are non-linear, generative, and parametric models of a certain visual phenomenon. The most frequent application of AAMs to date has been face modeling [Lanitis *et al.*, 1997]. However, AAMs may be useful for other phenomena too [Sclaroff and Isidoro, 1998, Jones and Poggio, 1998]. Typically an AAM is first fit to an image of a face; i.e. the model parameters are found to maximize the "match" between the model instance and the input image. The model parameters are then used in whatever the application is. For example, the parameters could be passed to a classifier to yield a face recognition algorithm. Many different classification tasks are possible. In [Lanitis *et al.*, 1997], for example, the same model was used for face recognition, pose estimation, and expression recognition. AAMs are a general-purpose image coding scheme, like Principal Components Analysis, but non-linear.

Fitting an AAM to an image is a non-linear optimization problem. The usual approach [Lanitis *et al.*, 1997, Cootes *et al.*, 1998, Cootes *et al.*, 2001, Cootes, 2001] is to iteratively solve for incremental *additive* updates to the parameters (the shape and appearance coefficients.) Given the current estimates of the shape parameters, it is possible to warp the input image backwards onto the model coordinate frame and then compute an error image between the current model instance and the image that the AAM is being fit to. In most previous algorithms, it is simply assumed that there is a *constant* linear relationship between this error image and the additive incremental updates to the parameters. The constant coefficients in this linear relationship can then be found either by linear regression [Lanitis *et al.*, 1997] or by other numerical methods [Cootes, 2001].

Unfortunately the assumption that there is such a simple relationship between the error image and the appropriate update to the model parameters is in general incorrect. See Section 2.3.3 for a counterexample. The result is that existing AAM fitting algorithms perform poorly, both in terms of the number of iterations required to converge, and in terms of the accuracy of the final fit. In this paper we propose a new analytical AAM fitting algorithm that does not make this simplifying assumption. Our algorithm in based on an extension to the *inverse compositional* image alignment algorithm [Baker and Matthews, 2001, Baker and Matthews, 2003]. The inverse compositional algorithm is only applicable to sets of warps that form a group. Unfortunately, the set of piecewise affine warps used in AAMs does not form a *group*. Hence, to use the inverse compositional algorithm, we derive first order approximations to the group operators of *composition* and *inversion*.

Using the inverse compositional algorithm also allows a different treatment of the appearance variation. Using the approach proposed in [Hager and Belhumeur, 1998], we project out the appearance variation thereby eliminating a great deal of computation. This modification is closely related to “shape AAMs” [Cootes and Kittipanya-ngam, 2002]. Another feature that we include is shape normalization. The linear shape variation of AAMs is often augmented by combining it with a 2D similarity transformation to “normalize” the shape. We show how the inverse compositional algorithm can be used to simultaneously fit the combination of the two warps (the linear AAM shape variation and a following global shape transformation, usually a 2D similarity transform.)

2 Linear Shape and Appearance Models: AAMs

Although they are perhaps the most well-known example, Active Appearance Models are just one instance in a large class of closely related *linear shape and appearance models* (and their associated fitting algorithms.) This class contains Active Appearance Models (AAMs) [Cootes *et al.*, 2001, Cootes *et al.*, 1998, Lanitis *et al.*, 1997], Shape AAMs [Cootes and Kittipanya-ngam, 2002], Active Blobs [Sclaroff and Isidoro, 1998], Morphable Models [Vetter and Poggio, 1997, Jones and Poggio, 1998, Blanz and Vetter, 1999], and Direct Appearance Models [Hou *et al.*, 2001], as well as possibly others. Many of these models were proposed independently in 1997-1998 [Lanitis *et al.*, 1997, Vetter and Poggio, 1997, Cootes *et al.*, 1998, Sclaroff and Isidoro, 1998, Jones and Poggio, 1998]. In this paper we use the term “Active Appearance Model” to refer generically to the entire class of linear shape and appearance models. We chose the term AAM rather than Active Blob or Morphable Model solely because it seems to have stuck better in the vision literature, not because the term was introduced earlier or because AAMs have any particular technical advantage. We also wanted to avoid introducing any new, and potentially confusing, terminology.

Unfortunately the previous literature is already very confusing. One thing that is particularly confusing is that the terminology often refers to the combination of a model and a fitting algorithm. For example, Active Appearance Models [Cootes *et al.*, 2001] strictly only refers to a specific model and an algorithm for fitting that model. Similarly, Direct Appearance Models [Hou *et al.*, 2001] refers to a different model-fitting algorithm pair. In order to simplify the terminology we will a make clear distinction between models and algorithms, *even though this sometimes means we will have to abuse previous terminology.* In particular, we use the term AAM to refer to the model, independent of the fitting algorithm. We also use AAM to refer to a slightly larger class of models than that described in [Cootes *et al.*, 2001]. We hope that this simplifies the situation.

In essence there are just two type of linear shape and appearance models, those which model shape and appearance independently, and those which parameterize shape and appearance with a single set of linear parameters. We refer to the first set as *independent linear shape and appearance models* and the second as *combined shape and appearance models*. Since the name AAM has stuck, we will also refer to the first set as *independent AAMs* and the second as *combined AAMs*.

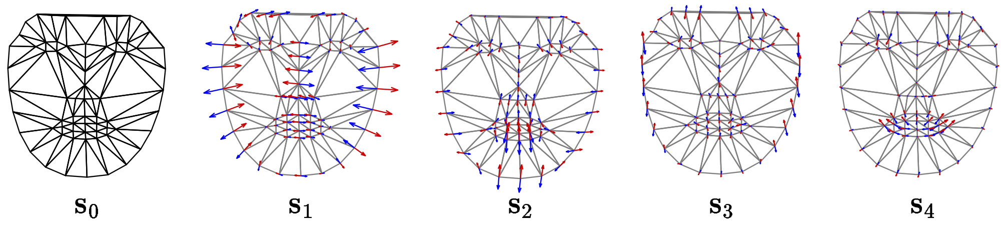


Figure 1: The linear shape model of an independent AAM. The model consists of a triangulated base mesh $\mathbf{s}_0$ plus a linear combination of n shape vectors $\mathbf{s}_i$. See Equation (2). Here we display the base mesh on the far left and to the right four shape vectors $\mathbf{s}_1$, $\mathbf{s}_2$, $\mathbf{s}_3$, and $\mathbf{s}_4$ overlayed on the base mesh.

2.1 Independent AAMs

2.1.1 Shape

As the name suggests, independent AAMs model shape and appearance separately. The *shape* of an independent AAM is defined by a mesh and in particular the vertex locations of the mesh. Usually the mesh is triangulated (although there are ways to avoid triangulating the mesh by using thin plate splines rather than piecewise affine warping [Cootes, 2001].) Mathematically, we define the shape $\mathbf{s}$ of an AAM as the coordinates of the v vertices that make up the mesh:

$$\mathbf{s} \; = \; (x_1, y_1, x_2, y_2, \ldots, x_v, y_v)^{\mathrm{T}}. \tag{1}$$

See Figure 1 for an example mesh. AAMs allow linear shape variation. This means that the shape $\mathbf{s}$ can be expressed as a base shape $\mathbf{s}_0$ plus a linear combination of n shape vectors $\mathbf{s}_i$:

$$\mathbf{s} \; = \; \mathbf{s}_0 + \sum_{i=1}^{n} p_i \mathbf{s}_i. \tag{2}$$

In this expression the coefficients p_i are the shape parameters. Since we can easily perform a linear reparameterization, wherever necessary we assume that the vectors $\mathbf{s}_i$ are orthonormal.

AAMs are normally computed from training data. The standard approach is to apply Principal Component Analysis (PCA) to the training meshes [Cootes *et al.*, 2001]. The base shape $\mathbf{s}_0$ is the mean shape and the vectors $\mathbf{s}_0$ are the n eigenvectors corresponding to the n largest eigenvalues.

An example independent AAM shape model is shown in Figure 1. On the left of the figure, we plot the triangulated base mesh $\mathbf{s}_0$. In the remainder of the figure, the base mesh $\mathbf{s}_0$ is overlayed with arrows corresponding to each of the first four shape vectors $\mathbf{s}_1$, $\mathbf{s}_2$, $\mathbf{s}_3$, and $\mathbf{s}_4$.

2.1.2 Appearance

The *appearance* of an independent AAM is defined within the base mesh $\mathbf{s}_0$. That way only pixels that are relevant to the phenomenon are modeled, and background pixels can be ignored. Let $\mathbf{s}_0$ also denote the set of pixels $\mathbf{x} = (x, y)^{\mathrm{T}}$ that lie inside the base mesh $\mathbf{s}_0$, a convenient abuse of

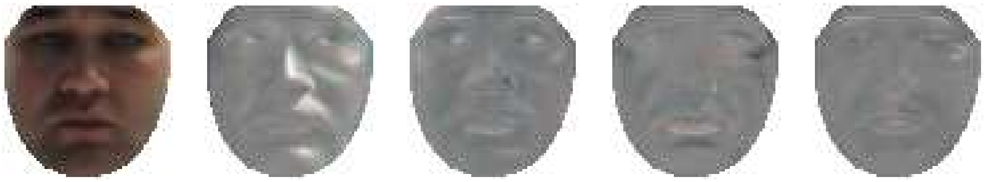


Figure 2: The linear appearance variation of an independent AAM. The model consists of a base appearance image A_0 defined on the pixels inside the base mesh $\mathbf{s}_0$ plus a linear combination of m appearance images A_i also defined on the same set of pixels. See Equation (3) for the formal definition of the model.

terminology. The appearance of an AAM is then an image $A(\mathbf{x})$ defined over the pixels $\mathbf{x} \in \mathbf{s}_0$. AAMs allow linear appearance variation. This means that the appearance $A(\mathbf{x})$ can be expressed as a base appearance $A_0(\mathbf{x})$ plus a linear combination of m appearance images $A_i(\mathbf{x})$:

$$A(\mathbf{x}) \;=\; A_0(\mathbf{x}) + \sum_{i=1}^{m} \lambda_i A_i(\mathbf{x}) \qquad \forall\, \mathbf{x} \in \mathbf{s}_0 \tag{3}$$

In this expression the coefficients λ_i are the appearance parameters. Since we can easily perform a linear reparameterization, wherever necessary we assume that the images A_i are orthonormal.

As with the shape component, the base appearance A_0 and the appearance images A_i are normally computed by applying Principal Component Analysis to a set of shape normalized training images [Cootes *et al.*, 2001]. The base appearance is set to be the mean image and the images A_i to be the m eigenimages corresponding to the m largest eigenvalues. The appearance of an example independent AAM is shown in Figure 2. On the left of the figure we plot the base appearance A_0. On the right we plot the first four appearance images A_1, A_2, A_3, and A_4.

2.1.3 Model Instantiation

Equations (2) and (3) describe the AAM shape and appearance variation. However, they do not describe how to generate a model instance. Given the AAM shape parameters $\mathbf{p} = (p_1, p_2, \ldots, p_n)^{\mathrm{T}}$ we can use Equation (2) to generate the shape of the AAM $\mathbf{s}$. Similarly, given the AAM appearance parameters $\boldsymbol{\lambda} = (\lambda_1, \lambda_2, \ldots, \lambda_m)^{\mathrm{T}}$, we can generate the AAM appearance $A(\mathbf{x})$ defined in the interior of the base mesh $\mathbf{s}_0$. The AAM model instance with shape parameters $\mathbf{p}$ and appearance parameters $\boldsymbol{\lambda}$ is then created by warping the appearance A from the base mesh $\mathbf{s}_0$ to the model shape $\mathbf{s}$. This process is illustrated in Figure 3 for concrete values of $\mathbf{p}$ and $\boldsymbol{\lambda}$.

In particular, the pair of meshes $\mathbf{s}_0$ and $\mathbf{s}$ define a piecewise affine warp from $\mathbf{s}_0$ to $\mathbf{s}$. For each triangle in $\mathbf{s}_0$ there is a corresponding triangle in $\mathbf{s}$. Any pair of triangles defines a unique affine warp from one to the other such that the vertices of the first triangle map to the vertices of the second triangle. See Section 4.1.1 for more details. The complete warp is then computed: (1) for any pixel $\mathbf{x}$ in $\mathbf{s}_0$ find out which triangle it lies in, and then (2) warp $\mathbf{x}$ with the affine warp for that triangle. We denote this piecewise affine warp $\mathbf{W}(\mathbf{x};\mathbf{p})$. The final AAM model instance is then

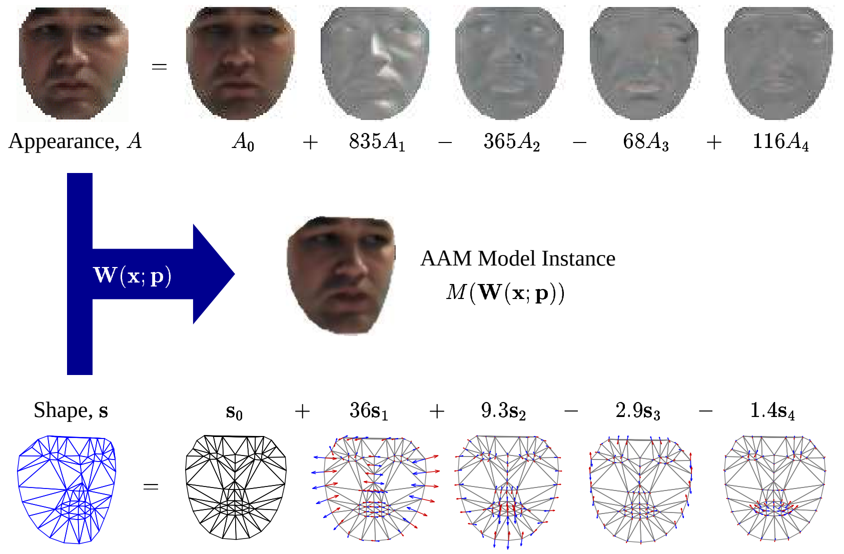

Appearance, A
=
A_0 + $835A_1$ − $365A_2$ − $68A_3$ + $116A_4$
$\mathbf{W}(\mathbf{x};\mathbf{p})$
AAM Model Instance
$M(\mathbf{W}(\mathbf{x};\mathbf{p}))$
Shape, $\mathbf{s}$
$\mathbf{s}_0$ + $36\mathbf{s}_1$ + $9.3\mathbf{s}_2$ − $2.9\mathbf{s}_3$ − $1.4\mathbf{s}_4$
=

Figure 3: An example of AAM instantiation. The shape parameters $\mathbf{p} = (p_1, p_2, \ldots, p_n)^{\mathrm{T}}$ are used to compute the model shape $\mathbf{s}$ and the appearance parameters $\boldsymbol{\lambda} = (\lambda_1, \lambda_2, \ldots, \lambda_m)^{\mathrm{T}}$ are used to compute the model appearance A. The model appearance is defined in the base mesh $\mathbf{s}_0$. 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 final AAM model instance, denoted $M(\mathbf{W}(\mathbf{x};\mathbf{p}))$, is computed by forwards warping the appearance A from $\mathbf{s}_0$ to $\mathbf{s}$ using $\mathbf{W}(\mathbf{x};\mathbf{p})$.

computed by forwards warping the appearance A from $\mathbf{s}_0$ to $\mathbf{s}$ with warp $\mathbf{W}(\mathbf{x};\mathbf{p})$. This process is defined by the following equation:

$$M(\mathbf{W}(\mathbf{x};\mathbf{p})) \;=\; A(\mathbf{x}). \tag{4}$$

This equation, and the forwards warping it denotes, should be interpreted as follows. Given a pixel $\mathbf{x}$ in $\mathbf{s}_0$, the destination of this pixel under the warp is $\mathbf{W}(\mathbf{x};\mathbf{p})$. The AAM model M at pixel $\mathbf{W}(\mathbf{x};\mathbf{p})$ in $\mathbf{s}$ is set to the appearance $A(\mathbf{x})$. Implementing this forwards warping to generate the model instance M without holes (see Figure 3) is actually somewhat tricky (and is best performed by backwards warping with the inverse warp from $\mathbf{s}$ to $\mathbf{s}_0$.) Fortunately, however, in the AAM fitting algorithms, only backwards warping onto the base mesh $\mathbf{s}_0$ is needed. Finally, note that the piecewise affine warping described in this section could be replaced with any other warp of interpolating between the mesh vertices. For example, thin plate splines could be used instead [Cootes, 2001]. In this paper we use piecewise affine warping since it is more efficient.

2.2 Combined AAMs

Independent AAMs have separate shape $\mathbf{p}$ and appearance $\boldsymbol{\lambda}$ parameters. On the other hand, combined AAMs just use a single set of parameters $\mathbf{c} = (c_1, c_2, \ldots, c_l)^{\mathrm{T}}$ to parameterized shape:

$$\mathbf{s} \;=\; \mathbf{s}_0 + \sum_{i=1}^{l} c_i \mathbf{s}_i \tag{5}$$

and appearance:

$$A(\mathbf{x}) \ = \ A_0(\mathbf{x}) + \sum_{i=1}^{l} c_i A_i(\mathbf{x}). \tag{6}$$

The shape and appearance parts of the model are therefore coupled. This coupling has a number of disadvantages. For example, it means that we can no longer assume the vectors $\mathbf{s}_i$ and $A_i(\mathbf{x})$ are respectively orthonormal. It also makes fitting more difficult. See the discussion at the end of this paper. On the other hand, combined AAMs have a number of advantages. First, the combined formulation is more general and is a strict superset of the independent formulation. To see this, set $\mathbf{c} = (p_1, p_2, \ldots, p_n, \lambda_1, \lambda_2, \ldots, \lambda_m)^{\mathrm{T}}$ and choose $\mathbf{s}_i$ and A_i appropriately. Second, combined AAMs often need less parameters to represent the same visual phenomenon to the same degree of accuracy; i.e. in practice $l \leq m + n$. Therefore fitting is more accurate and more efficient.

(This second advantage is actually not very significant. Since we will project out the appearance variation, as discussed in Section 4.2, the computational cost of our new algorithm is mainly just dependent on the number of shape parameters n and does not depend significantly on the number of appearance parameters m. The computational reduction by using l parameters rather than $n{+}m$ parameters is therefore non existent. For the same representational accuracy, $l \geq \max(n, m)$. Hence, our algorithm which uses independent AAMs and runs in time $\mathrm{O}(n)$ is actually more efficient than any for combined AAMs and which runs in time $\mathrm{O}(l)$.)

Combined AAMs are normally computed by taking an independent AAM and performing (a third) Principal Component Analysis on the training shape $\mathbf{p}$ and $\boldsymbol{\lambda}$ appearance parameters (weighted appropriately.) The shape and appearance parameters are then linearly reparameterized in terms of the new eigenvectors of the combined PCA. See [Cootes *et al.*, 2001] for the details, although note that the presentation there is somewhat different from the essentially equivalent presentation here. Also note that the definition of AAM in [Cootes *et al.*, 2001] only refers to our notion of combined AAMs. We use the expression AAM to refer to independent AAMs as well.

2.3 Fitting AAMs

2.3.1 Fitting Goal

Suppose we are given an input image $I(\mathbf{x})$ that we wish to fit an AAM to. What does this mean? Suppose for now that we know the optimal shape $\mathbf{p}$ and appearance $\boldsymbol{\lambda}$ parameters in the fit. This means that the image $I(\mathbf{x})$ and the model instance $M(\mathbf{W}(\mathbf{x};\mathbf{p})) = A(\mathbf{x})$ must be similar. In order to define the fitting process, we must formally define the criterion to be optimized in the fitting process. Naturally, we want to minimize the error between $I(\mathbf{x})$ and $M(\mathbf{W}(\mathbf{x};\mathbf{p})) = A(\mathbf{x})$. There are two coordinate frames in which this error can be computed, the coordinate frame of the image I and the coordinate frame of the AAM. It turns out that the better choice is to use the coordinate frame of the AAM; i.e. the base mesh $\mathbf{s}_0$. Suppose $\mathbf{x}$ is a pixel in $\mathbf{s}_0$. The corresponding pixel in the input image I is $\mathbf{W}(\mathbf{x};\mathbf{p})$. At the pixel $\mathbf{x}$ the AAM has the appearance $A(\mathbf{x}) = A_0(\mathbf{x}) + \sum_{i=1}^{m} \lambda_i A_i(\mathbf{x})$. At the pixel $\mathbf{W}(\mathbf{x};\mathbf{p})$, the input image has the intensity $I(\mathbf{W}(\mathbf{x};\mathbf{p}))$. We minimize the sum of squares of the difference between these two quantities:

$$\sum_{\mathbf{x}\in\mathbf{s}_0} \left[A_0(\mathbf{x}) + \sum_{i=1}^{m} \lambda_i A_i(\mathbf{x}) - I(\mathbf{W}(\mathbf{x};\mathbf{p})) \right]^2 \tag{7}$$

where the sum is performed over all pixels $\mathbf{x}$ in the base mesh $\mathbf{s}_0$. The goal of AAM fitting is then to minimize the expression in Equation (7) simultaneously with respect to the shape parameters $\mathbf{p}$ and the appearance parameters $\boldsymbol{\lambda}$. In general the optimization is nonlinear in the shape parameters $\mathbf{p}$, although linear in the appearance parameters $\boldsymbol{\lambda}$.

Before we describe any fitting algorithms, we introduce one piece of terminology. Denote:

$$E(\mathbf{x}) \;=\; A_0(\mathbf{x}) + \sum_{i=1}^{m} \lambda_i A_i(\mathbf{x}) - I(\mathbf{W}(\mathbf{x};\mathbf{p})) \tag{8}$$

to be the *error image*. The error image is defined in the coordinate frame of the AAM and can be computed as follows. For each pixel $\mathbf{x}$ in the base mesh $\mathbf{s}_0$, we compute the corresponding pixel $\mathbf{W}(\mathbf{x};\mathbf{p})$ in the input image by warping $\mathbf{x}$ with the piecewise affine warp $\mathbf{W}$. The input image I is then sampled at the pixel $\mathbf{W}(\mathbf{x};\mathbf{p})$; typically it is bilinearly interpolated at this pixel. The resulting value is then subtracted from the appearance $A_0(\mathbf{x}) + \sum_{i=1}^{m} \lambda_i A_i(\mathbf{x})$ at that pixel and the result stored in E. In essence, the input image I is backwards warped onto the base mesh $\mathbf{s}_0$ with warp $\mathbf{W}$ and then subtracted from the current AAM appearance $A_0(\mathbf{x}) + \sum_{i=1}^{m} \lambda_i A_i(\mathbf{x})$.

2.3.2 Inefficient Gradient Descent Algorithms

Perhaps the most natural way of minimizing the expression in Equation (7) is to use a standard gradient descent optimization algorithm. Various researchers have tried this. For example, Levenberg-Marquardt was used in [Sclaroff and Isidoro, 1998] and a stochastic gradient descent algorithm was used in [Jones and Poggio, 1998, Blanz and Vetter, 1999]. The advantage of these algorithms is that they use a principled, analytical algorithm, the convergence properties of which are well understood. The disadvantage of these gradient descent algorithms is that they are very slow. The partial derivatives, Hessian, and gradient direction all need to be recomputed in each iteration.

2.3.3 Efficient Ad-Hoc Fitting Algorithms

Because all previous gradient descent algorithms are so slow, a considerable amount of effort has been devoted in the past to developing other fitting algorithms that are more efficient [Cootes *et al.*, 2001, Cootes *et al.*, 1998, Sclaroff and Isidoro, 1998]. In all of these algorithms, the approach is to assume that there is a *constant* linear relationship between the error image $E(\mathbf{x})$ and *additive*

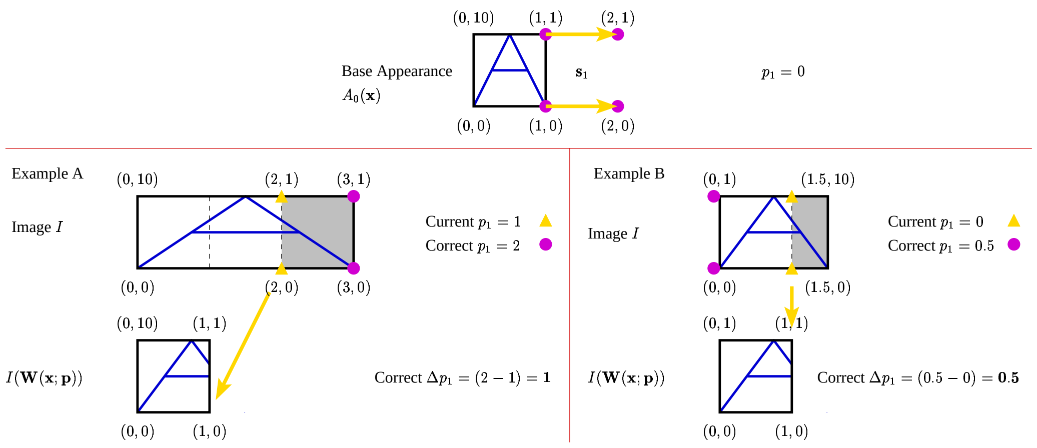


Figure 4: A counterexample showing that the linear relationship between Δp_i and the error image $E(\mathbf{x})$ does not have constant coeffi cients. We exhibit a simple AAM and two input images which have the same error image, but yet the correct update Δp_1 to the parameter p_1 is different in the two cases.

increments to the shape and appearance parameters:

$$\Delta p_i = \sum_{\mathbf{x} \in \mathbf{s}_0} R_i(\mathbf{x}) E(\mathbf{x}) \tag{9}$$

and:

$$\Delta \lambda_i = \sum_{\mathbf{x} \in \mathbf{s}_0} S_i(\mathbf{x}) E(\mathbf{x}) \tag{10}$$

where $R_i(\mathbf{x})$ and $S_i(\mathbf{x})$ are constant images defined on the base mesh $\mathbf{s}_0$. Here, constant means that $R_i(\mathbf{x})$ and $S_i(\mathbf{x})$ do not depend on p_i or λ_i. This assumption is motivated by the fact that almost all previous gradient descent algorithms boil down to computing Δp_i and $\Delta\lambda_i$ as linear functions of the error image and then updating $p_i \leftarrow p_i + \Delta p_i$ and $\lambda_i \leftarrow \lambda_i + \Delta\lambda_i$. In previous gradient descent algorithms, however, the equivalent of $R_i(\mathbf{x})$ and $S_i(\mathbf{x})$ are not constant, but instead depend on the AAM model parameters $\mathbf{p}$ and $\boldsymbol{\lambda}$. It is because they depend on the model parameters that they have to be recomputed. In essence, this is why the gradient descent algorithms are so inefficient.

To improve the efficiency, previous AAM fitting algorithms such as [Cootes *et al.*, 2001, Cootes *et al.*, 1998, Sclaroff and Isidoro, 1998] have either explicitly or implicitly simply assumed that $R_i(\mathbf{x})$ and $S_i(\mathbf{x})$ do not depend on the model parameters. This assumption is most definitely an approximation, as we now show. To provide a counterexample showing that $R_i(\mathbf{x})$ and $S_i(\mathbf{x})$ are not constant, it is sufficient to exhibit two cases where the error image is the same and for which different increments to the parameters should be applied. Figure 4 illustrates such a scenario.

The base appearance $A_0(\mathbf{x})$ is as presented at the top of the figure. The one and only shape vector $\mathbf{s}_1$ moves the two vertices on the right hand side of the square one unit to the right. The first example is Example A. The input image I consists of a stretched version of the base template. In this example $p_1 = 2$. The current estimate of $p_1 = 1$. When I is warped back onto the template

$I(\mathbf{W}(\mathbf{x};\mathbf{p}))$ is as shown in the figure. The correct estimate of Δp_1 should be equal to $2 - 1 = 1$. The second example, Example B, is similar. In this case the base template is stretched in the same direction, but this time by not as far. In this case, $p_1 = 0.5$. Suppose the current estimate of $p_1 = 0$. Then, when I is warped back onto the template $I(\mathbf{W}(\mathbf{x};\mathbf{p}))$ is exactly the same as in Example A. Hence, the error image is exactly the same. The correct estimate of Δp_1 in Example B should be $0.5 - 0 = 0.5$ however. Since this is different from the correct estimate of Δp_1 in Example A, this is a counterexample which shows that in general $R_i(\mathbf{x})$ is not a constant.

Note that although in this counterexample the difference between the two cases could be explained by a global affine warp (see Section 4.3), other counterexamples can easily be provided where a global correction does not help. Similarly, although in this counter example the direction of Δp_i is correct and it is just the magnitude that is wrong, other counterexamples can be provided where there the error images are the same, but the directions of the Δp_i are different.

Although the assumption that $R_i(\mathbf{x})$ and $S_i(\mathbf{x})$ are constant is incorrect, previous algorithms have set out to estimate them in a number of different ways. The original AAM formulation [Cootes *et al.*, 1998] estimated the update functions $R_i(\mathbf{x})$ and $S_i(\mathbf{x})$ by systematically perturbing the model parameters Δp_i and λ_i and recording the corresponding error image $E(\mathbf{x})$. The values of $R_i(\mathbf{x})$ and $S_i(\mathbf{x})$ are then estimate by linear regression. Later, the same authors proposed a finite difference approach [Cootes *et al.*, 2001,Cootes, 2001] that is essentially the same as the *difference decomposition* algorithm used in [Sclaroff and Isidoro, 1998].

(Note that the author of the original "difference decomposition" paper [Gleicher, 1997] may have been aware of the need to use difference decomposition in the *compositional* approach described in Section 3.2. It is hard to tell. However, the use of difference decomposition in [Sclaroff and Isidoro, 1998] makes the constant linear assumption in Equation (24) of that paper.)

3 Efficient Gradient Descent Image Alignment

As described above, existing AAM fitting algorithms fall into one of two categories. Either they take the analytical, gradient descent approach, with all the advantages of using a principled algorithm, but are very slow, or they make a provably incorrect assumption to obtain efficiency and in the process forfeit fitting accuracy. Is there not any way to use an efficient image alignment algorithm such as [Hager and Belhumeur, 1998]? The algorithm in [Hager and Belhumeur, 1998] unfortunately cannot be applied to piecewise affine warps; in fact it only applies to translations, 2D similarity transformations, affine warps, and a small collection of other esoteric warps.

Is there no other efficient gradient descent algorithm? The argument in Section 2.3.3 shows that there cannot be any efficient algorithm that solves for $\Delta\mathbf{p}$ and then updates the parameters $\mathbf{p} \leftarrow \mathbf{p} + \Delta\mathbf{p}$. Fortunately, this is not the only way to update the parameters however. Instead it is possible to update the entire warp by composing the current warp with the computed incremental warp with parameters $\Delta\mathbf{p}$. In particular, it is possible to update:

$$\mathbf{W}(\mathbf{x};\mathbf{p}) \;\leftarrow\; \mathbf{W}(\mathbf{x};\mathbf{p}) \circ \mathbf{W}(\mathbf{x};\Delta\mathbf{p}). \tag{11}$$

This *compositional* approach is different, yet provably equivalent, to the usual *additive* approach [Baker and Matthews, 2003]. It turns out that by extending this compositional approach we can derive an efficient gradient descent algorithm for independent AAMs as we now show.

3.1 Lucas-Kanade Image Alignment

The goal of image alignment is to find the location of a constant template image in an input image. The application of gradient descent to image alignment was first described in [Lucas and Kanade, 1981]. We now briefly review Lucas-Kanade image alignment. The goal of Lucas-Kanade is to find the locally "best" alignment by minimizing the sum of squares difference between a constant template image, $A_0(\mathbf{x})$ say, and an example image $I(\mathbf{x})$ with respect to the warp parameters $\mathbf{p}$:

$$\sum_{\mathbf{x}}[A_0(\mathbf{x}) - I(\mathbf{W}(\mathbf{x};\mathbf{p}))]^2. \tag{12}$$

Note the similarity with Equation (7). As in Section 2 above, $\mathbf{W}(\mathbf{x};\mathbf{p})$ is a warp that maps the pixels $\mathbf{x}$ from the template (i.e. the base mesh) image to the input image and has parameters $\mathbf{p}$. Note that $I(\mathbf{W}(\mathbf{x};\mathbf{p}))$ is an image with the same dimensions as the template; it is the input image I warped backwards onto the same coordinate frame as the template.

Solving for $\mathbf{p}$ is a nonlinear optimization problem, even if $\mathbf{W}(\mathbf{x};\mathbf{p})$ is linear in $\mathbf{p}$ because, in general, the pixels values $I(\mathbf{x})$ are nonlinear in (and essentially unrelated to) the pixel coordinates $\mathbf{x}$. To linearize the problem, the Lucas-Kanade algorithm assumes that an initial estimate of $\mathbf{p}$ is known and then iteratively solves for increments to the parameters $\Delta\mathbf{p}$; i.e. minimize:

$$\sum_{\mathbf{x}}[A_0(\mathbf{x}) - I(\mathbf{W}(\mathbf{x};\mathbf{p}+\Delta\mathbf{p}))]^2 \tag{13}$$

with respect to $\Delta\mathbf{p}$ and then update $\mathbf{p} \leftarrow \mathbf{p}+\Delta\mathbf{p}$. The expression in Equation (13) can be linearized about $\mathbf{p}$ using a Taylor series expansion to give:

$$\sum_{\mathbf{x}}\left[A_0(\mathbf{x}) - I(\mathbf{W}(\mathbf{x};\mathbf{p})) - \nabla I\frac{\partial\mathbf{W}}{\partial\mathbf{p}}\Delta\mathbf{p}\right]^2, \tag{14}$$

where ∇I is the *gradient* of the image evaluated at $\mathbf{W}(\mathbf{x};\mathbf{p})$, and $\frac{\partial\mathbf{W}}{\partial\mathbf{p}}$ is the *Jacobian* of the warp

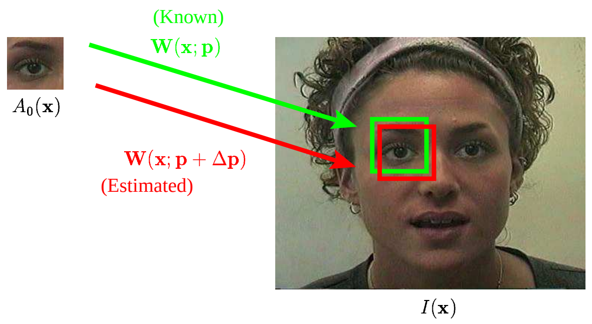

(Known)
$\mathbf{W}(\mathbf{x};\mathbf{p})$
$A_0(\mathbf{x})$
$\mathbf{W}(\mathbf{x};\mathbf{p}+\Delta\mathbf{p})$
(Estimated)
$I(\mathbf{x})$

Figure 5: A schematic overview of the Lucas-Kanade (forwards-additive) image alignment algorithm. Given current estimates of the parameters $\mathbf{p}$, Lucas-Kanade linearizes the problem and solves for incremental updates to the parameters $\Delta\mathbf{p}$ that are then added to the current estimates $\mathbf{p} \leftarrow \mathbf{p} + \Delta\mathbf{p}$.

evaluated at $\mathbf{p}$. The closed form solution of Equation (14) for $\Delta\mathbf{p}$ is:

$$\Delta\mathbf{p} \;=\; \mathbf{H}^{-1}\sum_{\mathbf{x}}\left[\nabla I\frac{\partial\mathbf{W}}{\partial\mathbf{p}}\right]^{\mathrm{T}}\left[A_0(\mathbf{x}) - I(\mathbf{W}(\mathbf{x};\mathbf{p}))\right] \tag{15}$$

where $\mathbf{H}$ is the Gauss-Newton approximation to the *Hessian* matrix:

$$\mathbf{H} = \sum_{\mathbf{x}}\left[\nabla I\frac{\partial\mathbf{W}}{\partial\mathbf{p}}\right]^{\mathrm{T}}\left[\nabla I\frac{\partial\mathbf{W}}{\partial\mathbf{p}}\right]. \tag{16}$$

An overview of the Lucas-Kanade algorithm is shown in Figure 5. In [Baker and Matthews, 2003] we refer to this as the *forwards-additive* algorithm. The additive part comes from the iterative update of the warp parameters: $\Delta\mathbf{p}$ is added each time. The forwards part denotes the direction of the warp parameter estimation: the warp projects *into* the image coordinate frame.

The Lucas-Kanade algorithm is slow. In general, both ∇I and $\frac{\partial\mathbf{W}}{\partial\mathbf{p}}$ depend on $\mathbf{p}$. Hence the Hessian and $\left[\nabla I\frac{\partial\mathbf{W}}{\partial\mathbf{p}}\right]^{\mathrm{T}}$ need to be recomputed in every iteration, both of which are slow operations.

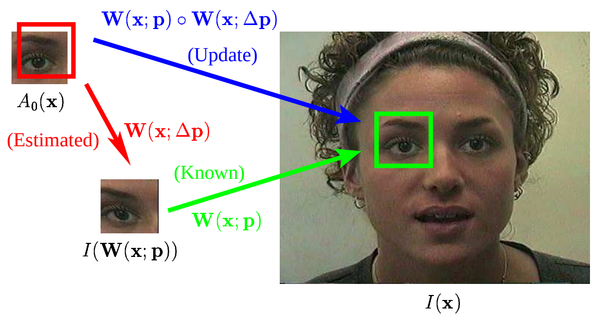
3.2 Forwards Compositional Image Alignment

In the Lucas-Kanade algorithm the warp parameters are computed by estimating a $\Delta\mathbf{p}$ offset from the current warp parameters $\mathbf{p}$. The *compositional* framework computes an *incremental warp* $\mathbf{W}(\mathbf{x};\Delta\mathbf{p})$ to be composed with the current warp $\mathbf{W}(\mathbf{x};\mathbf{p})$. The minimization is over:

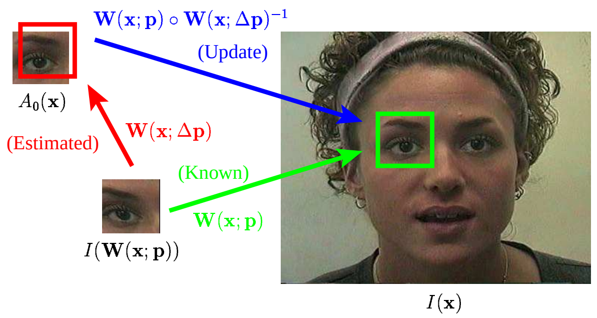
$$\sum_{\mathbf{x}}\left[A_0(\mathbf{x}) - I(\mathbf{W}(\mathbf{W}(\mathbf{x};\Delta\mathbf{p});\mathbf{p}))\right]^2, \tag{17}$$

and the update step involves *composing* the incremental and current warp:

$$\mathbf{W}(\mathbf{x};\mathbf{p}) \;\leftarrow\; \mathbf{W}(\mathbf{x};\mathbf{p})\circ\mathbf{W}(\mathbf{x};\Delta\mathbf{p}). \tag{18}$$


$\mathbf{W}(\mathbf{x};\mathbf{p}) \circ \mathbf{W}(\mathbf{x};\Delta\mathbf{p})$
(Update)
$A_0(\mathbf{x})$
(Estimated)
$\mathbf{W}(\mathbf{x};\Delta\mathbf{p})$
(Known)
$\mathbf{W}(\mathbf{x};\mathbf{p})$
$I(\mathbf{W}(\mathbf{x};\mathbf{p}))$
$I(\mathbf{x})$

(a) Forwards Compositional


$\mathbf{W}(\mathbf{x};\mathbf{p}) \circ \mathbf{W}(\mathbf{x};\Delta\mathbf{p})^{-1}$
(Update)
$A_0(\mathbf{x})$
(Estimated)
$\mathbf{W}(\mathbf{x};\Delta\mathbf{p})$
(Known)
$\mathbf{W}(\mathbf{x};\mathbf{p})$
$I(\mathbf{W}(\mathbf{x};\mathbf{p}))$
$I(\mathbf{x})$

(b) Inverse Compositional

Figure 6: (a) A schematic overview of the forwards-compositional image alignment algorithm. Given current estimates of the parameters, the forwards compositional algorithm solves for an incremental warp $\mathbf{W}(\mathbf{x};\Delta\mathbf{p})$ rather than a simple update to the parameters $\Delta\mathbf{p}$. The incremental warp is then composed with the current estimate of the warp: $\mathbf{W}(\mathbf{x};\mathbf{p}) \leftarrow \mathbf{W}(\mathbf{x};\mathbf{p}) \circ \mathbf{W}(\mathbf{x};\Delta\mathbf{p})$. The main advantage of the forwards compositional algorithm is that the alignment is computed between $A_0(\mathbf{x})$ and $I(\mathbf{W}(\mathbf{x};\mathbf{p}))$ starting from $\mathbf{p} = \mathbf{0}$. The Jacobian $\frac{\partial \mathbf{W}}{\partial \mathbf{p}}$ is therefore always evaluated at $\mathbf{p} = \mathbf{0}$ and so is constant across iterations. (b) A schematic overview of the inverse-compositional image alignment algorithm. The roles of $I(\mathbf{W}(\mathbf{x};\mathbf{p}))$ and $A_0(\mathbf{x})$ are reversed and the incremental warp $\mathbf{W}(\mathbf{x};\Delta\mathbf{p})$ is estimated in the other (inverse) direction. The incremental warp therefore has to be inverted before it is composed with the current estimate of the warp: $\mathbf{W}(\mathbf{x};\mathbf{p}) \leftarrow \mathbf{W}(\mathbf{x};\mathbf{p}) \circ \mathbf{W}(\mathbf{x};\Delta\mathbf{p})^{-1}$. The main advantage of the inverse compositional algorithm is that the Hessian and $\nabla A_0 \frac{\partial \mathbf{W}}{\partial \mathbf{p}}$ are both constant and so can be precomputed. Since this is most of the computation, the resulting algorithm is very efficient. See Figure 7 for the details of the algorithm.

If we compute the solution for $\Delta\mathbf{p}$ in Equation (17) then we have computed the incremental warp in the "image" direction. It can be composed with the current warp using Equation (18) and results in the *forwards compositional* algorithm [Baker and Matthews, 2003], also used in [Shum and Szeliski, 2000]. Taking the Taylor series expansion of Equation (17) gives:

$$\sum_{\mathbf{x}} \left[A_0(\mathbf{x}) - I(\mathbf{W}(\mathbf{W}(\mathbf{x};\mathbf{0});\mathbf{p})) - \nabla I(\mathbf{W}(\mathbf{x};\mathbf{p})) \frac{\partial \mathbf{W}}{\partial \mathbf{p}} \Delta\mathbf{p} \right]^2 . \tag{19}$$

At this point we assume that $\mathbf{p} = 0$ is the identity warp; i.e. $\mathbf{W}(\mathbf{x};\mathbf{0}) = \mathbf{x}$. There are then two differences between Equation (19) and Equation (14). First, the gradient is computed on $I(\mathbf{W}(\mathbf{x};\mathbf{p}))$. Second, the Jacobian is evaluated at $(\mathbf{x};\mathbf{0})$ and so is a constant that can be precomputed. The composition update step is computationally more costly than the update step for an additive algorithm, but this is offset by not having to compute the Jacobian $\frac{\partial \mathbf{W}}{\partial \mathbf{p}}$ in each iteration.

The key point in the forwards compositional algorithm, illustrated in Figure 6(a), is that the update is computed with respect to $\mathbf{p} = \mathbf{0}$ each time. This is why the Jacobian is constant.

3.3 Inverse Compositional Image Alignment

The *inverse compositional* algorithm is a modification of the forwards compositional algorithm where the roles of the template and example image are reversed. Rather than computing the incremental warp with respect to $I(\mathbf{W}(\mathbf{x};\mathbf{p}))$ it is computed with respect to the template $A_0(\mathbf{x})$.

The proof that this role reversal step results in an equivalent algorithm can be found in [Baker and Matthews, 2001, Baker and Matthews, 2003]. The intuitive reason is that when we reverse the roles of the images (in the compositional case), we just estimate the incremental warp in the other "inverse" direction. See Figure 6(b) for an overview of the inverse compositional algorithm.

Reversing the roles of $I(\mathbf{W}(\mathbf{x};\mathbf{p}))$ and $A_0(\mathbf{x})$ in Equation (17) results in the inverse compositional algorithm minimizing:

$$\sum_{\mathbf{x}} \left[I(\mathbf{W}(\mathbf{x};\mathbf{p})) - A_0(\mathbf{W}(\mathbf{x};\Delta\mathbf{p})) \right]^2 , \tag{20}$$

with respect to $\Delta\mathbf{p}$ and then updating the warp using:

$$\mathbf{W}(\mathbf{x};\mathbf{p}) \;\leftarrow\; \mathbf{W}(\mathbf{x};\mathbf{p}) \circ \mathbf{W}(\mathbf{x};\Delta\mathbf{p})^{-1}. \tag{21}$$

Taking the Taylor series expansion of Equation (20) gives:

$$\sum_{\mathbf{x}} \left[I(\mathbf{W}(\mathbf{x};\mathbf{p}) - A_0(\mathbf{W}(\mathbf{x};\mathbf{0})) - \nabla A_0 \frac{\partial \mathbf{W}}{\partial \mathbf{p}} \Delta\mathbf{p}) \right]^2 . \tag{22}$$

Assuming again that $\mathbf{W}(\mathbf{x};\mathbf{0})$ is the identity warp, the solution to this least squares problem is:

$$\Delta\mathbf{p} \;=\; \mathbf{H}^{-1} \sum_{\mathbf{x}} \left[\nabla A_0 \frac{\partial \mathbf{W}}{\partial \mathbf{p}} \right]^{\mathrm{T}} \left[I(\mathbf{W}(\mathbf{x};\mathbf{p})) - A_0(\mathbf{x}) \right] \tag{23}$$

where $\mathbf{H}$ is Hessian matrix with I replaced by A_0:

$$\mathbf{H} = \sum_{\mathbf{x}} \left[\nabla A_0 \frac{\partial \mathbf{W}}{\partial \mathbf{p}} \right]^{\mathrm{T}} \left[\nabla A_0 \frac{\partial \mathbf{W}}{\partial \mathbf{p}} \right] . \tag{24}$$

Since the template A_0 is constant and the Jacobian $\frac{\partial \mathbf{W}}{\partial \mathbf{p}}$ is always evaluated at $\mathbf{p} = \mathbf{0}$, most of the computation in Equations (23) and (24) can be moved to a precomputation step. The result is a very efficient image alignment algorithm. See Figure 7 for the details of the algorithm. Most of the computationally demanding steps are performed only once in a pre-computation step. The main algorithm just iterates: image warping (Step 1), image differencing (Step 2), image dot products (Step 7), multiplication with the precomputed inverse of the Hessian (Step 8), and update to the warp (Step 9). All of these steps can be implemented very efficiently. Note that Steps 1, 2, and 7 parallel equivalent steps in the efficient ad-hoc AAM fitting algorithms. The only additional computation is Steps 8 and 9 which are very efficient. These two steps essentially correct for the current estimates of the parameters $\mathbf{p}$ and avoid the problem illustrated in Figure 4.

The Inverse Compositional Algorithm

```
Pre-compute:
    (3) Evaluate the gradient $\nabla A_0$ of the template $A_0(\mathbf{x})$
    (4) Evaluate the Jacobian $\frac{\partial \mathbf{W}}{\partial \mathbf{p}}$ at $(\mathbf{x};\mathbf{0})$
    (5) Compute the steepest descent images $\nabla A_0 \frac{\partial \mathbf{W}}{\partial \mathbf{p}}$
    (6) Compute the Hessian matrix using Equation (24)

Iterate Until Converged:
    (1) Warp $I$ with $\mathbf{W}(\mathbf{x};\mathbf{p})$ to compute $I(\mathbf{W}(\mathbf{x};\mathbf{p}))$
    (2) Compute the error image $I(\mathbf{W}(\mathbf{x};\mathbf{p})) - A_0(\mathbf{x})$
    (7) Compute $\sum_{\mathbf{x}} [\nabla A_0 \frac{\partial \mathbf{W}}{\partial \mathbf{p}}]^{\mathrm{T}} [I(\mathbf{W}(\mathbf{x};\mathbf{p})) - T(\mathbf{x})]$
    (8) Compute $\Delta\mathbf{p}$ using Equation (23)
    (9) Update the warp $\mathbf{W}(\mathbf{x};\mathbf{p}) \leftarrow \mathbf{W}(\mathbf{x};\mathbf{p}) \circ \mathbf{W}(\mathbf{x};\Delta\mathbf{p})^{-1}$
```

Figure 7: The inverse compositional algorithm [Baker and Matthews, 2003]. All of the computationally demanding steps are performed in a pre-computation step. The main algorithm simply consists of image warping (Step 1), image differencing (Step 2), image dot products (Step 7), multiplication with the inverse of the Hessian (Step 8), and the update to the warp (Step 9). All of these steps can be implemented effi ciently.

4 Applying the Inverse Compositional Algorithm to AAMs

We now show how the inverse compositional algorithm can be applied to independent AAMs. The algorithm does not apply to combined AAMs. See Section 6.2 for more discussion of why not.

4.1 Application Without Appearance Variation

We first describe how the algorithm applies without any appearance variation; i.e. when $m = 0$. Comparing Equation (7) with Equation (12) we see that if there is no appearance variation, the inverse compositional algorithm applies as is. Examining Figure 7 we find that most of the steps in the algorithm are standard vector, matrix, and image operations such as computing image gradients and image differences. The only non-standard steps are: Step 1 warping I with the piecewise affine warp $\mathbf{W}(\mathbf{x};\mathbf{p})$, Step 4 computing the Jacobian of the piecewise affine warp, and Step 9 inverting the incremental piecewise affine warp and composing it with the current estimate of the piecewise affine warp. We now describe how each of these steps is performed.

4.1.1 Piecewise Affine Warping

The image $I(\mathbf{W}(\mathbf{x};\mathbf{p}))$ is computed by backwards warping the input image I with the warp $\mathbf{W}(\mathbf{x};\mathbf{p})$; i.e. for each pixel $\mathbf{x}$ in the base mesh $\mathbf{s}_0$ we compute $\mathbf{W}(\mathbf{x};\mathbf{p})$ and sample (bilinearly interpolate) the image I at that location. Every pixel $\mathbf{x}$ in the base mesh $\mathbf{s}_0$ lies in a triangle. Suppose that the vertices of that triangle are $(x_i^0, y_i^0)^{\mathrm{T}}$, $(x_j^0, y_j^0)^{\mathrm{T}}$, and $(x_k^0, y_k^0)^{\mathrm{T}}$. Also suppose that the vertices of the corresponding triangle in the AAM mesh are $(x_i, y_i)^{\mathrm{T}}$, $(x_j, y_j)^{\mathrm{T}}$, and $(x_k, y_k)^{\mathrm{T}}$.

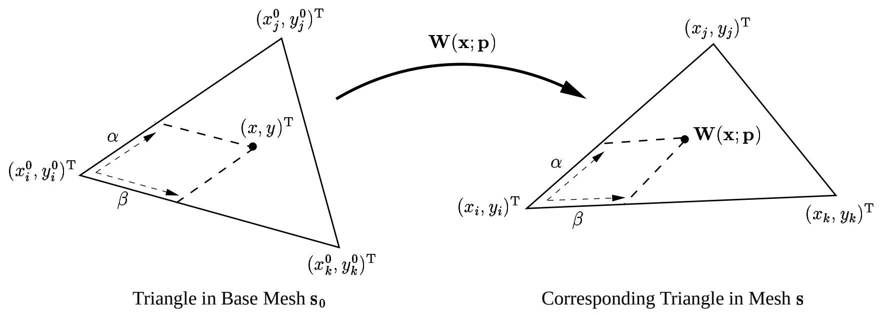

$(x_j^0, y_j^0)^{\mathrm{T}}$
$\mathbf{W}(\mathbf{x};\mathbf{p})$
$(x_j, y_j)^{\mathrm{T}}$
$(x, y)^{\mathrm{T}}$
α
$(x_i^0, y_i^0)^{\mathrm{T}}$
β
$(x_k^0, y_k^0)^{\mathrm{T}}$
$\mathbf{W}(\mathbf{x};\mathbf{p})$
α
$(x_i, y_i)^{\mathrm{T}}$
β
$(x_k, y_k)^{\mathrm{T}}$
Triangle in Base Mesh $\mathbf{s}_0$
Corresponding Triangle in Mesh $\mathbf{s}$

Figure 8: Computing the piecewise affi ne warp $\mathbf{W}(\mathbf{x};\mathbf{p})$. Each pixel $\mathbf{x} = (x, y)^{\mathrm{T}}$ in the base mesh $\mathbf{s}_0$ lies in a triangle. The pixel $(x, y)^{\mathrm{T}}$ can be decomposed into one vertex plus α times a vector down one side of the triangle plus β times a vector down the other side of the triangle. The destination of $(x, y)^{\mathrm{T}}$ under the piecewise affi ne warp $\mathbf{W}(\mathbf{x};\mathbf{p})$ is the equivalent expression for the other triangle in mesh $\mathbf{s}$.

These vertices can be computed from the shape parameters $\mathbf{p}$ using Equation (2).

One way to implement the piecewise affine warp is illustrated in Figure 8. Consider the pixel $\mathbf{x} = (x, y)^{\mathrm{T}}$ in the triangle $(x_i^0, y_i^0)^{\mathrm{T}}$, $(x_j^0, y_j^0)^{\mathrm{T}}$, and $(x_k^0, y_k^0)^{\mathrm{T}}$ in the base mesh $\mathbf{s}_0$. This pixel can be uniquely expressed as:

$$\mathbf{x} \; = \; (x, y)^{\mathrm{T}} \; = \; (x_i^0, y_i^0)^{\mathrm{T}} + \alpha\left[(x_j^0, y_j^0)^{\mathrm{T}} - (x_i^0, y_i^0)^{\mathrm{T}}\right] + \beta\left[(x_k^0, y_k^0)^{\mathrm{T}} - (x_i^0, y_i^0)^{\mathrm{T}}\right] \tag{25}$$

where:

$$\alpha \; = \; \frac{(x - x_i^0)(y_k^0 - y_i^0) - (y - y_i^0)(x_k^0 - x_i^0)}{(x_j^0 - x_i^0)(y_k^0 - y_i^0) - (y_j^0 - y_i^0)(x_k^0 - x_i^0)} \tag{26}$$

and:

$$\beta \; = \; \frac{(y - y_i^0)(x_j^0 - x_i^0) - (x - x_i^0)(y_j^0 - y_i^0)}{(y_j^0 - y_i^0)(x_k^0 - x_i^0) - (x_j^0 - x_i^0)(y_k^0 - y_i^0)}. \tag{27}$$

The result of applying the piecewise affine warp is then:

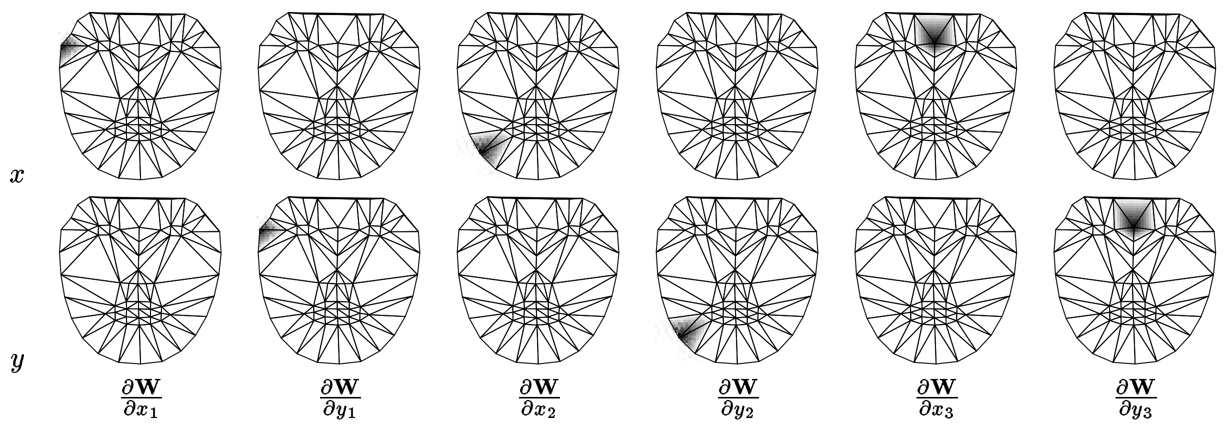
$$\mathbf{W}(\mathbf{x};\mathbf{p}) \; = \; (x_i, y_i)^{\mathrm{T}} + \alpha\left[(x_j, y_j)^{\mathrm{T}} - (x_i, y_i)^{\mathrm{T}}\right] + \beta\left[(x_k, y_k)^{\mathrm{T}} - (x_i, y_i)^{\mathrm{T}}\right] \tag{28}$$

where $(x_i, y_i)^{\mathrm{T}}$, $(x_j, y_j)^{\mathrm{T}}$, and $(x_k, y_k)^{\mathrm{T}}$ are the vertices of the corresponding triangle in $\mathbf{s}$. Together, Equation (26), (27), and (28) constitute a simple affine warp:

$$\mathbf{W}(\mathbf{x};\mathbf{p}) \; = \; (a_1 + a_2 \cdot x + a_3 \cdot y, a_4 + a_5 \cdot x + a_6 \cdot y)^{\mathrm{T}}. \tag{29}$$

The 6 parameters of the this warp $(a_1, a_2, a_3, a_4, a_5, a_6)$ can easily be computed from the shape parameters $\mathbf{p}$ by combing Equations (2), (26), (27), and (28). This computation only needs to be performed once per triangle, not once per pixel. To implement the piecewise affine warp efficiently, the computation should be structured:

- Given $\mathbf{p}$ compute $(x_i, y_i)^{\mathrm{T}}$ for all vertices in $\mathbf{s}$.



x
y
$\frac{\partial \mathbf{W}}{\partial x_1}$
$\frac{\partial \mathbf{W}}{\partial y_1}$
$\frac{\partial \mathbf{W}}{\partial x_2}$
$\frac{\partial \mathbf{W}}{\partial y_2}$
$\frac{\partial \mathbf{W}}{\partial x_3}$
$\frac{\partial \mathbf{W}}{\partial y_3}$

Figure 9: The Jacobians $\frac{\partial \mathbf{W}}{\partial x_i}$ and $\frac{\partial \mathbf{W}}{\partial y_i}$ with respect to the vertices of the mesh $\mathbf{s}$ for 3 different vertices i. The x component of the Jacobian is in the top row and the y component is in the bottom row.

- Compute $(a_1, a_2, a_3, a_4, a_5, a_6)$ for each triangle.
- For each pixel $\mathbf{x}$ in the mesh $\mathbf{s}_0$, lookup the triangle that $\mathbf{x}$ lies in and then lookup the corresponding values of $(a_1, a_2, a_3, a_4, a_5, a_6)$. Finally compute $\mathbf{W}(\mathbf{x}; \mathbf{p})$ using Equation (29).

If we raster scan the mesh $\mathbf{s}_0$ we can avoid looking up $(a_1, a_2, a_3, a_4, a_5, a_6)$ most of the time by creating a lookup table for the triangle identity that codes when the triangle identity changes.

4.1.2 Computing the Warp Jacobian

The destination of the pixel $\mathbf{x}$ under the piecewise affine warp $\mathbf{W}(\mathbf{x}; \mathbf{p})$ depends on the AAM shape parameters $\mathbf{p}$ through the vertices of the mesh $\mathbf{s}$. From Equation (1) remember that these vertices are denoted: $\mathbf{s} = (x_1, y_1, x_2, y_2, \ldots, x_v, y_v)^{\mathrm{T}}$. Applying the chain rule to the warp $\mathbf{W}(\mathbf{x}; \mathbf{p})$ gives:

$$\frac{\partial \mathbf{W}}{\partial \mathbf{p}} = \sum_{i=1}^{v} \left[\frac{\partial \mathbf{W}}{\partial x_i} \frac{\partial x_i}{\partial \mathbf{p}} + \frac{\partial \mathbf{W}}{\partial y_i} \frac{\partial y_i}{\partial \mathbf{p}} \right]. \tag{30}$$

The first components of the Jacobian are $\frac{\partial \mathbf{W}}{\partial x_i}$ and $\frac{\partial \mathbf{W}}{\partial y_i}$, the Jacobians of the warp with respect to the vertices of the mesh $\mathbf{s}$. From Equation (28) we see that:

$$\frac{\partial \mathbf{W}}{\partial x_i} = (1 - \alpha - \beta, 0)^{\mathbf{T}} \quad \text{and} \quad \frac{\partial \mathbf{W}}{\partial y_i} = (0, 1 - \alpha - \beta)^{\mathbf{T}}. \tag{31}$$

These Jacobians are images the size of the base mesh $\mathbf{s}_0$. Examples of these Jacobians are included in Figure 9. Each Jacobian is the Jacobian with respect to a particular vertex. The Jacobian $\frac{\partial \mathbf{W}}{\partial x_i}$ denotes the rate of change of the destination of the warp $\mathbf{W}(\mathbf{x}; \mathbf{p})$ with respect to the vertex x_i. As can be seen, the Jacobian is only non-zero in the triangles around x_i. It takes the maximum value of 1 at the vertex x_i and decays away linearly as given by Equation (31). The second components

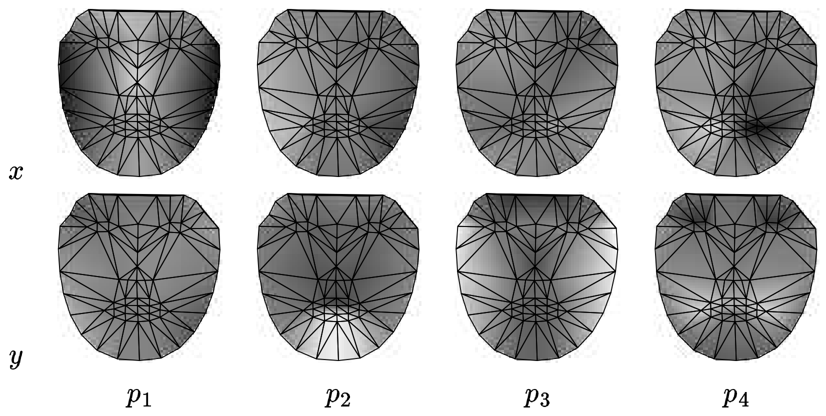


Figure 10: The Jacobian $\frac{\partial \mathbf{W}}{\partial \mathbf{p}}$ corresponding to the four shape vectors in Figure 1. The first mode p_1 mostly corresponds to a left-right rotation of the face, the second mode p_2 to the lowering of the chin, the third mode p_3 to the up-down rotation of the head, and the fourth mode p_4 to the opening of the mouth.

of the Jacobian are $\frac{\partial x_i}{\partial \mathbf{p}}$ and $\frac{\partial y_i}{\partial \mathbf{p}}$. Differentiating Equation (2) gives:

$$\frac{\partial x_i}{\partial \mathbf{p}} = (\mathbf{s}_1^{x_i}, \mathbf{s}_1^{x_i}, \ldots, \mathbf{s}_v^{x_i}) \quad \text{and} \quad \frac{\partial y_i}{\partial \mathbf{p}} = (\mathbf{s}_1^{y_i}, \mathbf{s}_1^{y_i}, \ldots, \mathbf{s}_v^{y_i}) \tag{32}$$

where $\mathbf{s}_j^{x_i}$ denotes the component of $\mathbf{s}_j$ that corresponds to x_i and similarly for y_i. The quantities $\frac{\partial x_i}{\partial \mathbf{p}}$ and $\frac{\partial y_i}{\partial \mathbf{p}}$ are therefore just the shape vectors $\mathbf{s}_i$ rearranged appropriately.

Putting together the components in Equations (31–32) results in the overall Jacobian $\frac{\partial \mathbf{W}}{\partial \mathbf{p}}$ looking like those in Figure 10. The array of 2×4 (base mesh $\mathbf{s}_0$ shaped) images correspond to the Jacobian for the four shape vectors in Figure 1. In particular, the first pair of images mostly correspond to a left-right rotation of the face, the second pair to the lowering of the chin, the third pair to the up-down rotation of the head, and the fourth pair to the opening of the mouth.

4.1.3 Warp Inversion

In Step 9 of the inverse compositional algorithm we must invert the incremental piecewise affine warp $\mathbf{W}(\mathbf{x}; \Delta\mathbf{p})$ to compute $\mathbf{W}(\mathbf{x}; \Delta\mathbf{p})^{-1}$. Since:

$$\mathbf{W}(\mathbf{x}; \Delta\mathbf{p}) = \mathbf{W}(\mathbf{x}; \mathbf{0}) + \frac{\partial \mathbf{W}}{\partial \mathbf{p}} \Delta\mathbf{p} = \mathbf{x} + \frac{\partial \mathbf{W}}{\partial \mathbf{p}} \Delta\mathbf{p} + \mathrm{O}(\Delta\mathbf{p}^2) \tag{33}$$

(remember that $\mathbf{W}(\mathbf{x}; \mathbf{0}) = \mathbf{x}$ is the identity warp) we therefore have:

$$\mathbf{W}(\mathbf{x}; \Delta\mathbf{p}) \circ \mathbf{W}(\mathbf{x}; -\Delta\mathbf{p}) = \mathbf{x} - \frac{\partial \mathbf{W}}{\partial \mathbf{p}} \Delta\mathbf{p} + \frac{\partial \mathbf{W}}{\partial \mathbf{p}} \Delta\mathbf{p} = \mathbf{x} + \mathrm{O}(\Delta\mathbf{p}^2). \tag{34}$$

It therefore follows that to first order in $\Delta\mathbf{p}$:

$$\mathbf{W}(\mathbf{x}; \Delta\mathbf{p})^{-1} = \mathbf{W}(\mathbf{x}; -\Delta\mathbf{p}). \tag{35}$$

Note that the two Jacobians in Equation (34) are not evaluated at exactly the same location, but since they are evaluated at points $O(\Delta\mathbf{p})$ apart, they are equal to zeroth order in $\Delta\mathbf{p}$. Since the difference is multiplied by $\Delta\mathbf{p}$ we can ignore the first and higher order terms. Also note that the composition of two warps is not strictly defined and so the argument in Equation (34) is informal. The essence of the argument is correct, however. Once we have the derived the first order approximation to the composition of two piecewise affine warps below, we can then use that definition of composition in the argument above. The result is that the warp $\mathbf{W}(\mathbf{x}; -\Delta\mathbf{p})$ followed by the warp $\mathbf{W}(\mathbf{x}; \Delta\mathbf{p})$ is equal to the identity warp to first order in $\Delta\mathbf{p}$.

4.1.4 Composing the Incremental Warp with the Current Warp Estimate

After we have inverted the piecewise affine warp $\mathbf{W}(\mathbf{x}; \Delta\mathbf{p})$ to compute $\mathbf{W}(\mathbf{x}; \Delta\mathbf{p})^{-1}$ in Step 9 we must compose the result with the current warp $\mathbf{W}(\mathbf{x}; \mathbf{p})$ to obtain $\mathbf{W}(\mathbf{x}; \mathbf{p}) \circ \mathbf{W}(\mathbf{x}; \Delta\mathbf{p})^{-1}$. Given the current estimate of the parameters $\mathbf{p}$ we can compute the current mesh vertex locations $\mathbf{s} = (x_1, y_1, \ldots, x_v, y_v)^{\mathrm{T}}$ using Equation (2). From the section above, we know that the parameters of $\mathbf{W}(\mathbf{x}; \Delta\mathbf{p})^{-1}$ are $-\Delta\mathbf{p}$. Given these parameters, we can use Equation (2) again to estimate the corresponding changes to the base mesh vertex locations:

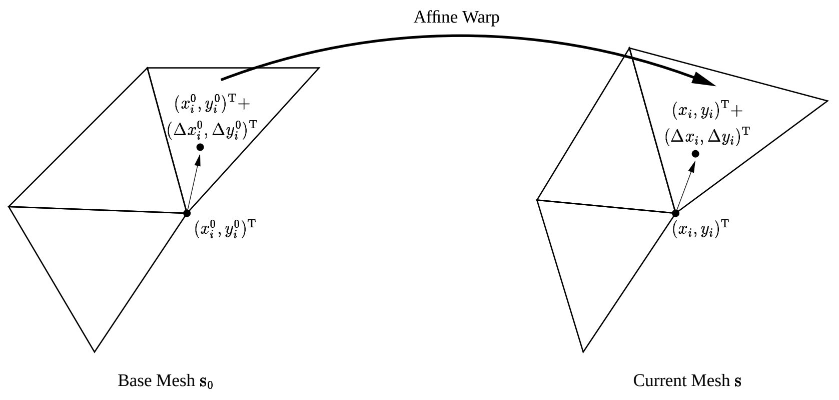
$$\Delta\mathbf{s}_0 \;=\; -\sum_{i=1}^{n} \Delta p_i \mathbf{s}_i \tag{36}$$

where $\Delta\mathbf{s}_0 = (\Delta x_1^0, \Delta y_1^0, \ldots, \Delta x_v^0, \Delta y_v^0)^{\mathrm{T}}$ are the changes to the base mesh vertex locations corresponding to $\mathbf{W}(\mathbf{x}; \Delta\mathbf{p})^{-1}$. We now need to compose $\mathbf{W}(\mathbf{x}; \Delta\mathbf{p})^{-1}$ with $\mathbf{W}(\mathbf{x}; \mathbf{p})$. In order to do this, we must compute the corresponding changes to the vertices in the current mesh vertex locations $\Delta\mathbf{s} = (\Delta x_1, \Delta y_1, \ldots, \Delta x_v, \Delta y_v)^{\mathrm{T}}$. Once we know these locations we can then compute the parameters of $\mathbf{W}(\mathbf{x}; \mathbf{p}) \circ \mathbf{W}(\mathbf{x}; \Delta\mathbf{p})^{-1}$ by applying Equation (2) in the other direction:

$$p_i' \;=\; \mathbf{s}_i \cdot (\mathbf{s} + \Delta\mathbf{s} - \mathbf{s}_0) \tag{37}$$

where p_i' is the i^{th} parameter of $\mathbf{W}(\mathbf{x}; \mathbf{p}) \circ \mathbf{W}(\mathbf{x}; \Delta\mathbf{p})^{-1}$, $\cdot$ denotes vector dot product, and we have assumed that the shape vectors $\mathbf{s}_i$ are orthonormal. The situation is then as illustrated in Figure 11.

All that remains to be described is how $\Delta\mathbf{s}$ is computed from $\Delta\mathbf{s}_0$. Consider the i^{th} vertex in the mesh. We therefore need to compute $(\Delta x_i, \Delta y_i)^{\mathrm{T}}$ from $(\Delta x_i^0, \Delta y_i^0)^{\mathrm{T}}$. Now consider any of the mesh triangles that contains the i^{th} vertex. For this triangle there is an affine warp between the base mesh $\mathbf{s}_0$ and the current mesh $\mathbf{s}$. See Section 4.1.1 for more details. One way to compute $(\Delta x_i, \Delta y_i)^{\mathrm{T}}$ from $(\Delta x_i^0, \Delta y_i^0)^{\mathrm{T}}$ is to apply the piecewise affine warp for that triangle to $(x_i^0, y_i^0)^{\mathrm{T}} + (\Delta x_i^0, \Delta y_i^0)^{\mathrm{T}}$ to obtain $(x_i, y_i)^{\mathrm{T}} + (\Delta x_i, \Delta y_i)^{\mathrm{T}}$. The only problem with this approach is which triangle do we use? Using a different triangle will mean using a different affine warp and so



Affine Warp
$(x_i^0, y_i^0)^{\rm T}+$
$(\Delta x_i^0, \Delta y_i^0)^{\rm T}$
$(x_i^0, y_i^0)^{\rm T}$
$(x_i, y_i)^{\rm T}+$
$(\Delta x_i, \Delta y_i)^{\rm T}$
$(x_i, y_i)^{\rm T}$
Base Mesh $\mathbf{s}_0$
Current Mesh $\mathbf{s}$

Figure 11: Composing the incremental warp $\mathbf{W}(\mathbf{x};\Delta\mathbf{p})^{-1}$ with the current warp $\mathbf{W}(\mathbf{x};\Delta\mathbf{p})$. We know the current mesh $\mathbf{s}$ and incremental updates to the base mesh $(\Delta x_i^0, \Delta y_i^0)^{\rm T}$. We need to compute incremental updates to the current mesh $(\Delta x_i, \Delta y_i)^{\rm T}$. This can be performed by applying the affi ne warp for each triangle about the $i^{\rm th}$ vertex to $(x_i^0, y_i^0)^{\rm T} + (\Delta x_i^0, \Delta y_i^0)^{\rm T}$ to obtain multiple estimates of $(x_i, y_i)^{\rm T} + (\Delta x_i, \Delta y_i)^{\rm T}$. In general these estimates will differ because the affi ne warps for each triangle are different. In essence this is why composing two piecewise affi ne warps is hard. We average the multiple estimates to compute the new mesh vertex locations. The new warp parameters can then be computed using Equation (37).

the destination $(x_i, y_i)^{\rm T} + (\Delta x_i, \Delta y_i)^{\rm T}$ will be different. In essence this is the reason that the composition of two piecewise affine warps is hard to define. In general there will be several triangles that share the $i^{\rm th}$ vertex. One possibility is to use the triangle that contains the point $(x_i^0, y_i^0)^{\rm T} + (\Delta x_i^0, \Delta y_i^0)^{\rm T}$. The problem with this approach is that that point could lie outside the base mesh $\mathbf{s}_0$. Instead we compute the destination $(x_i, y_i)^{\rm T} + (\Delta x_i, \Delta y_i)^{\rm T}$ for each triangle and average the results. This will tend to smooth the warp at each vertex, but that is desirable anyway.

4.2 Including Appearance Variation

We have now described all of the steps needed to apply the inverse compositional algorithm to an independent AAM assuming that there is no appearance variation. More generally, we wish to use the same algorithm to minimize the expression in Equation (7). We now describe how this can be done using the trick proposed in [Hager and Belhumeur, 1998]. Rewrite Equation (7) as:

$$\sum_{\mathbf{x}\in\mathbf{s}_0}\left[A_0(\mathbf{x}) + \sum_{i=1}^{m}\lambda_i A_i(\mathbf{x}) - I(\mathbf{W}(\mathbf{x};\mathbf{p}))\right]^2 = \left\| A_0(\mathbf{x}) + \sum_{i=1}^{m}\lambda_i A_i(\mathbf{x}) - I(\mathbf{W}(\mathbf{x};\mathbf{p}))\right\|^2 \quad (38)$$

where $\|\cdot\|$ is the L2 norm. This expression must be minimized simultaneously with respect to $\mathbf{p}$ and $\boldsymbol{\lambda} = (\lambda_1, \lambda_2, \ldots, \lambda_m)^{\rm T}$. If we denote the linear subspace spanned by a collection of vectors A_i by $\mathrm{span}(A_i)$ and its orthogonal complement by $\mathrm{span}(A_i)^\perp$ Equation (38) can be rewritten as:

$$\left\| A_0(\mathbf{x}) + \sum_{i=1}^{m}\lambda_i A_i(\mathbf{x}) - I(\mathbf{W}(\mathbf{x};\mathbf{p}))\right\|^2_{\mathrm{span}(A_i)^\perp} + \left\| A_0(\mathbf{x}) + \sum_{i=1}^{m}\lambda_i A_i(\mathbf{x}) - I(\mathbf{W}(\mathbf{x};\mathbf{p}))\right\|^2_{\mathrm{span}(A_i)} \quad (39)$$

where $\|\cdot\|_L^2$ denotes the square of the L2 norm of the vector projected into the linear subspace L. The first of the two terms immediately simplifies. Since the norm only considers the components of vectors in the orthogonal complement of $\mathrm{span}(A_i)$, any component in $\mathrm{span}(A_i)$ itself can be dropped. We therefore wish to minimize:

$$\| A_0(\mathbf{x}) - I(\mathbf{W}(\mathbf{x};\mathbf{p})) \|^2_{\mathrm{span}(A_i)^\perp} + \left\| A_0(\mathbf{x}) + \sum_{i=1}^{m} \lambda_i A_i(\mathbf{x}) - I(\mathbf{W}(\mathbf{x};\mathbf{p})) \right\|^2_{\mathrm{span}(A_i)} . \qquad (40)$$

The first of these two terms does not depend upon λ_i. For any $\mathbf{p}$, the minimum value of the second term is always 0. Therefore the minimum value can be found sequentially by first minimizing the first term with respect to $\mathbf{p}$ alone, and then using that optimal value of $\mathbf{p}$ as a constant to minimize the second term with respect to the λ_i. Assuming that the basis vectors A_i are orthonormal, the second minimization has a simple closed-form solution:

$$\lambda_i \;=\; \sum_{\mathbf{x}\in\mathbf{s}_0} A_i(\mathbf{x}) \cdot \left[I(\mathbf{W}(\mathbf{x};\mathbf{p})) - A_0(\mathbf{x}) \right], \qquad (41)$$

the dot product of A_i with the final error image obtained after doing the first minimization.

Minimizing the first term in Equation (40) is not really any different from applying the the inverse compositional algorithm to the AAM with no appearance variation. The only difference is that we need to work in linear subspace $\mathrm{span}(A_i)^\perp$ rather than in the full vector space defined over the pixels in $\mathbf{s}_0$. We do not even need to project the error image into this subspace. All we need to do is project $\nabla A_0 \frac{\partial \mathbf{W}}{\partial \mathbf{p}}$ into that subspace in Step 5 of the inverse compositional algorithm. See Figure 7. (The error image does not need to be projected into this subspace because Step 7 of the algorithm is really the dot product of the error image with $\nabla A_0 \frac{\partial \mathbf{W}}{\partial \mathbf{p}}$. So long as one of the two terms in a dot product is projected into a linear subspace, the result is the same as they both were.)

The steepest descent images $\nabla A_0 \frac{\partial \mathbf{W}}{\partial \mathbf{p}}$ can be projected into $\mathrm{span}(A_i)^\perp$ as follows:

$$\nabla A_0 \frac{\partial \mathbf{W}}{\partial \mathbf{p}} \;-\; \sum_{i=1}^{m} \left[\sum_{\mathbf{x}\in\mathbf{s}_0} A_i(\mathbf{x}) \cdot \nabla A_0 \frac{\partial \mathbf{W}}{\partial \mathbf{p}} \right] A_i(\mathbf{x}) \qquad (42)$$

See Figure 12 for a summary of the inverse compositional algorithm with appearance variation.

4.3 Including a Global Shape Transform

The most common way of constructing an AAM [Cootes *et al.*, 2001] consists of first "normalizing" the mesh so that it is as close as possible to the base mesh [Cootes *et al.*, 2001]. Typically, a 2D similarity transformation (translation, rotation, and scale) is used, although an affine warp, a homography, or any other global warp could be used instead. Because the training data is normal-

The Inverse Compositional Algorithm with Appearance Variation

```
Pre-compute:
    (3)  Evaluate the gradient $\nabla A_0$ of the template $A_0(\mathbf{x})$
    (4)  Evaluate the Jacobian $\frac{\partial \mathbf{W}}{\partial \mathbf{p}}$ at $(\mathbf{x};\mathbf{0})$
    (5)  Compute the modifi ed steepest descent images using Equation (42)
    (6)  Compute the Hessian matrix using modifi ed steepest descent images
Iterate:
    (1)  Warp $I$ with $\mathbf{W}(\mathbf{x};\mathbf{p})$ to compute $I(\mathbf{W}(\mathbf{x};\mathbf{p}))$
    (2)  Compute the error image $I(\mathbf{W}(\mathbf{x};\mathbf{p})) - A_0(\mathbf{x})$
    (7)  Compute dot product of modifi ed steepest descent images with error image
    (8)  Compute $\Delta\mathbf{p}$ by multiplying by inverse Hessian
    (9)  Update the warp $\mathbf{W}(\mathbf{x};\mathbf{p}) \leftarrow \mathbf{W}(\mathbf{x};\mathbf{p}) \circ \mathbf{W}(\mathbf{x};\Delta\mathbf{p})^{-1}$
Post-computation:
    (10) Compute $\lambda_i$ using Equation (41)
```

Figure 12: The inverse compositional algorithm with appearance variation. The only differences from the algorithm without appearance variation are: (1) the expression in Equation (42) is computed in place of the steepest descent images $\nabla A_0 \frac{\partial \mathbf{W}}{\partial \mathbf{p}}$ in Step 5, and (2) Step 10 is added to compute the appearance parameters.

ized in this way, the linear shape variation in the AAM does not model the translation, rotation, and scaling in the original training data. To avoid this problem, the linear shape variation is typically augmented with a global shape transformation in the following manner.

4.3.1 Adding a Global Shape Transform to an AAM

Suppose $\mathbf{N}(\mathbf{x};\mathbf{q})$ is a parameterized set of warps. For example, $\mathbf{N}(\mathbf{x};\mathbf{q})$ might be the set of 2D similarity transforms:

$$\mathbf{N}(\mathbf{x};\mathbf{q}) \;=\; (q_1, q_2)^{\mathrm{T}} + (1+q_3)\cdot(x\cos q_4 - y\sin q_4, x\sin q_4 + y\cos q_4)^{\mathrm{T}} \qquad (43)$$

where the four parameters $(q_1, q_2, q_3, q_4)^{\mathrm{T}}$ have the following interpretations. The first pair $(q_1, q_2)^{\mathrm{T}}$ is the translation. The third q_3 is related to the scale; the scale is $1 + q_3$ so that the scale is 1 when $q_3 = 0$. Finally, q_4 is the rotation, again set so that when $q_4 = 0$ there is no rotation. Although the set of 2D similarity transformation is the most common choice, any other global warp could be used. Other natural choices are affine warps and homographies [Bergen *et al.*, 1992].

Note that the above is not the only way to parameterize the set of 2D similarity transformations. Another way is to define the set as a special subset of the set of piecewise affine warps used in AAMs. Suppose the base mesh is $\mathbf{s}_0 = (x_1^0, y_1^0, \ldots, x_v^0, y_v^0)^{\mathrm{T}}$. If we choose $\mathbf{s}_1 = (1, 0, \cdots, 1, 0)^{\mathrm{T}}$, $\mathbf{s}_2 = (0, 1, \cdots, 0, 1)^{\mathrm{T}}$, $\mathbf{s}_3 = \mathbf{s}_0 = (x_1^0, y_1^0, \ldots, x_v^0, y_v^0)^{\mathrm{T}}$, and $\mathbf{s}_4 = (y_1^0, -x_1^0, \ldots, y_v^0, -x_v^0)^{\mathrm{T}}$, then the set of allowed linear AAM shape variation is exactly equal to the set of 2D similarity transformations. It is also straightforward to transform between the parameters of this linear parameterization and the parameters of the more common (nonlinear) parameterization in Equation (43).

Given $\mathbf{N}(\mathbf{x};\mathbf{q})$, the definition of an AAM is then augmented from Equation (4) to:

$$M(\mathbf{N}(\mathbf{W}(\mathbf{x};\mathbf{p});\mathbf{q})) \;=\; A(\mathbf{x}) \;=\; A_0(\mathbf{x}) + \sum_{i=1}^{m} \lambda_i A_i(\mathbf{x}). \tag{44}$$

Given appearance parameters $\boldsymbol{\lambda} = (\lambda_1, \lambda_2, \ldots, \lambda_m)^{\mathrm{T}}$, the AAM appearance $A(\mathbf{x})$ is generated in the base mesh $\mathbf{s}_0$. The model instance M is then created by warping the appearance A from the base mesh $\mathbf{s}_0$ first with the linear shape warp $\mathbf{W}(\mathbf{x};\mathbf{p})$ and then with the normalizing warp $\mathbf{N}(\mathbf{x};\mathbf{q})$. See Section 2.1.3 for more details. Fitting the AAM then consists of minimizing:

$$\sum_{\mathbf{x}\in\mathbf{s}_0} \left[A_0(\mathbf{x}) + \sum_{i=1}^{m} \lambda_i A_i(\mathbf{x}) - I(\mathbf{N}(\mathbf{W}(\mathbf{x};\mathbf{p});\mathbf{q})) \right]^2 \tag{45}$$

simultaneously with respect to the appearance parameters $\boldsymbol{\lambda}$, the linear shape parameters $\mathbf{p}$, and the global shape warp parameters $\mathbf{q}$.

Note that this new definition of an AAM in Equation (44) is different from just augmenting the linear shape variation to include a 2D similarity transformation. For example, we could add the four shape vectors $\mathbf{s}_1 = (1, 0, \cdots, 1, 0)^{\mathrm{T}}$, $\mathbf{s}_2 = (0, 1, \cdots, 0, 1)^{\mathrm{T}}$, $\mathbf{s}_3 = \mathbf{s}_0 = (x_1^0, y_1^0, \ldots, x_v^0, y_v^0)^{\mathrm{T}}$, and $\mathbf{s}_4 = (y_1^0, -x_1^0, \ldots, y_v^0, -x_v^0)^{\mathrm{T}}$ to the learnt AAM shape variation (and then orthonormalize.) The AAM would then be able to move under 2D similarity transformations, *as well as* the original linear AAM shape variation. This is not the same as moving under the linear shape variation *followed by* the 2D similarity transformation. For example, suppose the only AAM linear shape variation consists of “opening the mouth.” The shape vector then “moves” the mouth vertices up and down in the image. If we just augment the shape vectors with the 2D similarity shape vectors, the AAM will be able to translate, rotate, scale, and move the mouth vertices up and down. However, if the whole head is rotated by 90 degrees, opening the mouth corresponds to moving the mouth vertices left and right. This shape variation is only possible if the linear AAM shape variation is *followed by* a global shape transformation, as defined in Equation (44).

4.3.2 Fitting an AAM with a Global Shape Transform

We now briefly describe how the inverse compositional algorithm can be used to fit an AAM with a global shape transformation; i.e. how to apply the inverse compositional algorithm to the warp:

$$\mathbf{N} \circ \mathbf{W}(\mathbf{x};\mathbf{q},\mathbf{p}) \;=\; \mathbf{N}(\mathbf{W}(\mathbf{x};\mathbf{p});\mathbf{q}) \tag{46}$$

rather than the warp $\mathbf{W}(\mathbf{x};\mathbf{p})$. We just need to repeat the steps in Sections 4.1.1–4.1.4 and in Section 4.2. In this description we will assume that the shape normalizing warp $\mathbf{N}(\mathbf{x};\mathbf{q})$ is defined by the subset of the piecewise affine warp defined by the shape vectors $\mathbf{s}_1 = (1, 0, \cdots, 1, 0)^{\mathrm{T}}$, $\mathbf{s}_2 = (0, 1, \cdots, 0, 1)^{\mathrm{T}}$, $\mathbf{s}_3 = \mathbf{s}_0 = (x_1^0, y_1^0, \ldots, x_v^0, y_v^0)^{\mathrm{T}}$, and $\mathbf{s}_4 = (y_1^0, -x_1^0, \ldots, y_v^0, -x_v^0)^{\mathrm{T}}$ since this is slightly simpler. Using the nonlinear parameterization of Equation (43) is also possible.

Warping: The only thing that changes from Section 4.1.1 is how the destination mesh points are computed. Given the linear shape parameters $\mathbf{p}$, the destination of the mesh vertices under the linear shape variation $\mathbf{W}(\mathbf{x};\mathbf{p})$ can be computed using Equation (2). The destination of these vertices under the shape normalization $\mathbf{N}(\mathbf{x};\mathbf{q})$ is then computed as follows. Given the parameters $\mathbf{q}$, we use Equation (2) to compute the destination of 3 mesh vertices under the warp. We then solve for the parameters $(a_1, a_2, a_3, a_4, a_5, a_6)$ of the affine warp:

$$\mathbf{N}(\mathbf{x};\mathbf{q}) \;=\; (a_1 + a_2 \cdot x + a_3 \cdot y, a_4 + a_5 \cdot x + a_6 \cdot y)^{\mathrm{T}}. \tag{47}$$

We then apply this affine warp to the destination of the mesh vertices under $\mathbf{W}(\mathbf{x};\mathbf{p})$ to compute the destination under $\mathbf{N} \circ \mathbf{W}$. Once the destination of the mesh vertices under $\mathbf{N} \circ \mathbf{W}$ are known the piecewise affine warping is performed exactly as in Section 4.1.1.

Computing the Jacobian: Since $\mathbf{W}(\mathbf{x};\mathbf{0}) = \mathbf{N}(\mathbf{x};\mathbf{0}) = \mathbf{x}$, the identity warp, we have:

$$\frac{\partial}{\partial \mathbf{q}} \mathbf{N} \circ \mathbf{W} \;=\; \frac{\partial \mathbf{N}}{\partial \mathbf{q}} \tag{48}$$

and:

$$\frac{\partial}{\partial \mathbf{p}} \mathbf{N} \circ \mathbf{W} \;=\; \frac{\partial \mathbf{W}}{\partial \mathbf{p}}. \tag{49}$$

The computation of the Jacobian $(\frac{\partial}{\partial \mathbf{q}} \mathbf{N} \circ \mathbf{W}, \frac{\partial}{\partial \mathbf{p}} \mathbf{N} \circ \mathbf{W})$ is therefore exactly the same as in Section 4.1.2. We just compute separate Jacobians for $\mathbf{W}(\mathbf{x};\mathbf{p})$ and $\mathbf{N}(\mathbf{x};\mathbf{q})$ for the $\mathbf{p}$ and $\mathbf{q}$ parameters respectively. The overall Jacobian is then the concatenation of these Jacobians.

Warp Inversion: The inverse of $\mathbf{N} \circ \mathbf{W}(\mathbf{x};\Delta\mathbf{q},\Delta\mathbf{p})$ is $\mathbf{N} \circ \mathbf{W}(\mathbf{x};-\Delta\mathbf{q},-\Delta\mathbf{p})$ to first order in $\Delta\mathbf{q}$ and $\Delta\mathbf{p}$. The argument follows from the first order approximation:

$$\mathbf{N} \circ \mathbf{W}(\mathbf{x};\Delta\mathbf{q},\Delta\mathbf{p}) \;=\; \mathbf{x} + \frac{\partial \mathbf{N}}{\partial \mathbf{q}} \Delta\mathbf{q} + \frac{\partial \mathbf{W}}{\partial \mathbf{p}} \Delta\mathbf{p} \tag{50}$$

in the same way as in Section 4.1.3.

Warp Composition: There are a variety of ways of performing the composition of the two warps. Empirically we found it best to treat the two warps $\mathbf{N}$ and $\mathbf{W}$ separately. The new parameters of the warp $\mathbf{W}$ are computed in the same way as in Section 4.1.4. Equation (2) is used to compute the changes to the base mesh vertex locations $(\Delta x_i^0, \Delta y_i^0)^{\mathrm{T}}$ from $\Delta\mathbf{p}$. These are then converted to changes in the destination mesh vertex locations $(\Delta x_i, \Delta y_i)^{\mathrm{T}}$ by applying the affine warp of $\mathbf{W}(\mathbf{x};\mathbf{p})$ to each triangle and then averaging the results. The equivalent of Equation (37) is then used to compute the new values of $\mathbf{p}$.

The shape normalizing warp $\mathbf{N}$ is treated similarly, and independently. Equation (2) is used to compute the changes to the base mesh vertex locations $(\Delta x_i^0, \Delta y_i^0)^{\mathrm{T}}$ from $\Delta\mathbf{q}$. These are then converted to changes in the destination mesh vertex locations $(\Delta x_i, \Delta y_i)^{\mathrm{T}}$ by applying the global shape transform $\mathbf{N}(\mathbf{x};\mathbf{q})$. (Since $\mathbf{N}$ is a global transform, there is no need to treat each triangle separately and then average the results.) The equivalent of Equation (37) is then used to compute the new values of $\mathbf{q}$.

Appearance Variation: The treatment of appearance variation is exactly as in Section 4.2. Solving for the warp $\mathbf{N} \circ \mathbf{W}$ rather than for $\mathbf{W}$ doesn't change anything except that the steepest descent images for both $\mathbf{p}$ and $\mathbf{q}$ need to be projected into $\mathrm{span}(A_i)^\perp$ using Equation (42).

The inverse compositional AAM fitting algorithm with appearance variation and global shape transform is summarized in Figure 13. In Step 5 we compute the modified steepest descent images:

$$\nabla A_0 \frac{\partial \mathbf{N}}{\partial \mathbf{q}} - \sum_{i=1}^{m} \left[\sum_{\mathbf{x}\in\mathbf{s}_0} A_i(\mathbf{x}) \cdot \nabla A_0 \frac{\partial \mathbf{N}}{\partial \mathbf{q}} \right] A_i(\mathbf{x}) \tag{51}$$

for $\mathbf{q}$ and:

$$\nabla A_0 \frac{\partial \mathbf{W}}{\partial \mathbf{p}} - \sum_{i=1}^{m} \left[\sum_{\mathbf{x}\in\mathbf{s}_0} A_i(\mathbf{x}) \cdot \nabla A_0 \frac{\partial \mathbf{W}}{\partial \mathbf{p}} \right] A_i(\mathbf{x}) \tag{52}$$

for $\mathbf{p}$. The steepest descent images are then concatenated into a single vector with 4 images for $\mathbf{q}$ followed by n images for $\mathbf{p}$. If we denote the elements of this vector $\mathrm{SD}_i(\mathbf{x})$ for $i = 1, \ldots, n+4$, the $(i,j)^{\mathrm{th}}$ element of the $(n+4)\times(n+4)$ Hessian matrix is then computed in Step 6:

$$\mathbf{H} = \sum_{\mathbf{x}\in\mathbf{s}_0} \mathrm{SD}_i(\mathbf{x}) \cdot \mathrm{SD}_j(\mathbf{x}). \tag{53}$$

We compute the appearance parameters in Step (10) using:

$$\lambda_i = \sum_{\mathbf{x}\in\mathbf{s}_0} A_i(\mathbf{x}) \cdot \left[I(\mathbf{N}(\mathbf{W}(\mathbf{x};\mathbf{p});\mathbf{q})) - A_0(\mathbf{x}) \right]. \tag{54}$$

4.4 Other Extensions to the Algorithm

We have described how the inverse compositional image alignment algorithm can be applied to AAMs. The field of image alignment is well studied and over the years a number of extensions

Inverse Compositional Algorithm with Appearance Variation and Global Shape Transform

```
Pre-compute:
    (3)  Evaluate the gradient $\nabla A_0$ of the template $A_0(\mathbf{x})$
    (4)  Evaluate the Jacobians $\frac{\partial \mathbf{W}}{\partial \mathbf{p}}$ and $\frac{\partial \mathbf{N}}{\partial \mathbf{q}}$ at $(\mathbf{x}; \mathbf{0})$
    (5)  Compute the modifi ed steepest descent images using Equations (51) and (52)
    (6)  Compute the Hessian matrix using Equation (53)
Iterate:
    (1)  Warp $I$ with $\mathbf{W}(\mathbf{x}; \mathbf{p})$ followed by $\mathbf{N}(\mathbf{x}; \mathbf{q})$ to compute $I(\mathbf{N}(\mathbf{W}(\mathbf{x}; \mathbf{p}); \mathbf{q}))$
    (2)  Compute the error image $I(\mathbf{N}(\mathbf{W}(\mathbf{x}; \mathbf{p}); \mathbf{q})) - A_0(\mathbf{x})$
    (7)  Compute $\sum_{\mathbf{x} \in \mathbf{s}_0} \mathrm{SD}_i(\mathbf{x}) \cdot [I(\mathbf{N}(\mathbf{W}(\mathbf{x}; \mathbf{p}); \mathbf{q})) - A_0(\mathbf{x})]$ for $i = 1, \ldots, n+4$
    (8)  Compute $(\Delta\mathbf{q}, \Delta\mathbf{p})$ by multiplying the resulting vector by the inverse Hessian
    (9)  Update $(\mathbf{N} \circ \mathbf{W})(\mathbf{x}; \mathbf{q}, \mathbf{p}) \leftarrow (\mathbf{N} \circ \mathbf{W})(\mathbf{x}; \mathbf{q}, \mathbf{p}) \circ (\mathbf{N} \circ \mathbf{W})(\mathbf{x}; \Delta\mathbf{q}, \Delta\mathbf{p})^{-1}$
Post-computation:
    (10) Compute $\lambda_i$ using Equation (54)
```

Figure 13: The inverse compositional AAM fi tting algorithm with appearance variation and global shape transform. Most of the computation is similar to that in Figure 12 for the inverse compositional algorithm with appearance variation, except there are $n+4$ shape parameters, 4 in $\mathbf{q}$ and n in $\mathbf{p}$. We therefore compute $n+4$ modifi ed steepest descent images in Step 5, a $(n+4) \times (n+4)$ Hessian matrix in Step 6, and $n+4$ steepest descent parameter updates in Step 7. These updates are formed into a $n+4$ dimensional vector and multiplied by the inverse of the Hessian in Step 8 to give the $n+4$ dimensional vector $(\Delta\mathbf{q}, \Delta\mathbf{p})$.

and heuristics have been developed to improve the performance of the algorithms. Most of these can easily be applied to the algorithm that we have just described. Three examples include:

Hierarchical Processing: The fitting algorithm can be applied hierarchically on a Gaussian image pyramid to reduce the likelihood of falling into a local minimum [Bergen *et al.*, 1992].

Progressive Transformation Complexity: The fitting algorithm can be applied incrementally to more and more complex warps; i.e. first fit on a small number of the shape parameters $(p_1, \ldots, p_n)^{\mathrm{T}}$ and then incrementally add more and more complexity [Bergen *et al.*, 1992].

Levenberg-Marquardt: It is possible to use the Levenberg-Marquardt inverse compositional algorithm instead of the Gauss-Newton inverse compositional algorithm described in this paper. See [Baker and Matthews, 2003] for the details of that algorithm.

5 Empirical Evaluation

We have proposed a new fitting algorithm for AAMs. The performance of AAM fitting algorithms depends on a wide variety of factors. For example, it depends on whether hierarchical processing, progressive transformation complexity, and adaptive step-size algorithms such as Levenberg-Marquardt are used. See Section 4.4. The performance can also be very dependent on minor details such as the definition of the gradient filter used to compute ∇A_0. Comparing like with like is therefore very difficult. Another thing that makes empirical evaluation hard is the wide variety of AAM fitting algorithms [Cootes *et al.*, 2001, Cootes *et al.*, 1998, Sclaroff and Isidoro, 1998, Jones and Poggio, 1998, Blanz and Vetter, 1999] and the lack of a standard test set.

In our evaluation we take the following philosophy. Instead of comparing our algorithm with the original AAM fitting algorithm or any other algorithm (the results of which would have limited meaning), we set up a collection of systematic experiments where we only vary one component of the algorithm. In this paper, we have discussed three main changes to AAM fitting:

1. We use the inverse compositional warp update rather than the additive; i.e. we use Step 9 in Figure 13 to update the warp parameters rather than simply updating $\mathbf{p} \leftarrow \mathbf{p} + \Delta\mathbf{p}$.
2. We use an analytical derivation of the steepest descent images rather than a numerical approximation [Cootes *et al.*, 2001, Cootes, 2001, Sclaroff and Isidoro, 1998, Gleicher, 1997].
3. We project out the appearance variation as described in Section 4.2 rather than fitting for it as in [Cootes *et al.*, 2001, Cootes, 2001, Sclaroff and Isidoro, 1998, Gleicher, 1997].

Each of these changes can be made independently and so we evaluate each one independently. We compare our AAM fitting algorithm with and without each of these changes. For example, in the first experiment we try the two alternatives for Step 9 of the algorithm and compare them. Every other line of code in the implementation is always exactly the same.

5.1 Generating the Inputs

We begin by taking a video of a person moving their head (both rigidly and non-rigidly.) We select a random subset of the images, hand-mark feature points, and build an AAM. We then track the head through the sequence. Visually we check that the tracking and the reconstruction is good. Ideally we want to use the tracking results as ground-truth. This, however, is problematic because it biases the results towards the algorithm used to generate the ground truth. We avoid this problem by overlaying the reconstructed AAM over the original movie. We therefore have two movies. The original movie, and a second synthetic movie containing the background of the original movie and the face region synthetically generated from the AAM. By comparing the performance of the algorithms on these two movies we should be able to detect any bias in the ground-truth. Empirically we found almost no difference between the performance of any of the algorithms on the corresponding real and synthetic sequences and conclude there is no bias. For lack of space, we just present the results on the synthetic sequence. The results on the original sequence are almost identical.

Our experimental procedure consists of running each pair of algorithms on a collection of inputs, evaluating the results, and averaging. Each input consists of: (1) the AAM, (2) the initial shape, appearance, and normalization parameters, and (3) an image. Each input test case is generated as follows. Given any frame from the video, the ground-truth parameters for that image are then randomly perturbed to generate the initial estimates of the parameters to start the algorithms with. The shape parameters are randomly generated from independent Gaussian distributions with variance equal to the eigenvalue of that mode in the PCA performed during AAM construction. The normalizing warp (similarity) parameters are generated using the same procedure as in [Baker and Matthews, 2003]. Two distinguished points in the mesh are perturbed with Gaussian noise of a certain variance and the similarity parameters then solved for. The amount of perturbation is then sampled by varying the variance of the perturbations to the distinguished points. Finally, the appearance parameters are set to be the mean appearance. The results presented below are a result of averaging the performance over a fixed number of trials (specifically 20) for each of the 300 frames in the sequence.

5.2 The Evaluation Metrics

Given the initial parameters, the AAM fitting algorithm should hopefully converge to the ground-truth parameters. We measure this convergence in two ways, similarly to [Baker and Matthews, 2003]. The first measure is the *average rate of convergence.* We plot the RMS error in the mesh point locations against the iteration number of the algorithm. If the algorithm converges the RMS error should reduce to close to zero. We average these graphs (for approximately the same starting error) over all cases where all algorithms converge. We say that an algorithm converges if the RMS mesh point error is less than 1.0 pixels after 20 iterations. The second measure is the *average frequency of convergence.* We count the number of times each algorithm has converged (after 20 iterations), divide by the total number of trials, and multiply by 100 to convert to a percentage.

We do not measure the accuracy of the appearance parameters because once the shape parameters have been estimated, estimating the appearance parameters is a simple linear operation. See Equation (41). Also, comparing the appearance algorithms by their appearance estimates is not possible because the appearance is not computed until the "project out" algorithm has converged.

5.3 Experiment 1: The Update Rule

In our first experiment we compare the inverse compositional update in Step 9 of the algorithm with the usual additive update of: $\mathbf{p} \leftarrow \mathbf{p} + \Delta\mathbf{p}$. (Because the roles of the image and the template were switched in the derivation of the inverse compositional algorithm, we actually update $\mathbf{p} \leftarrow \mathbf{p} - \Delta\mathbf{p}$.) All previous efficient AAM fitting algorithms, including [Cootes *et al.*, 2001, Cootes *et al.*, 1998, Sclaroff and Isidoro, 1998] use the additive update. We present results both with and without a global similarity transform as described in Section 4.3. Without the global similarity transform, the similarity shape vectors are added as the first four shape vectors rather than being in a separate transform that is composed with the shape model. The results without the global similarity transform are included in Figure 14 and the results with it are included in Figure 15.

In the first row of each figure (a) and (b) we plot results just perturbing the shape parameters, in the second row (c) and (d) we plot results just perturbing the similarity parameters, and in (e) and (f) we plot results perturbing both. The weighting between the shape and similarity parameters is chosen so that they are roughly equivalent as measured by the frequency of convergence.

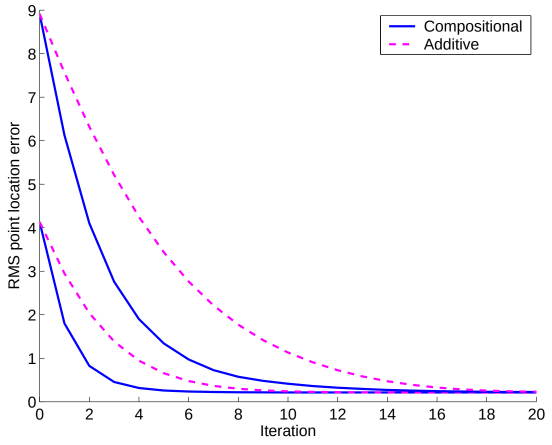
The first thing to notice from Figures 14 and 15 is that in almost all cases the inverse compositional update outperforms the additive. Either the rate of convergence is faster or the frequency of convergence is higher. The one notable exception is for the shape component of the results with the global similarity transform in Figures 15(a) and (b). In this case, the global similarity transform partially hides the problem described in Section 2.3.3. However, the problem is really just pushed into the similarity parameters and so the overall performance in Figures 15(e) and (f) is worse. These results also illustrate that the inverse compositional algorithm does not always outperform the additive algorithm. It just performs better in many scenarios, and similarly in others. This may be the reason that other authors have found cases where the algorithms perform similarly [Cootes and Kittipanya-ngam, 2002]. In either case, it is better to use the inverse compositional update.

The second thing to notice from Figures 14 and 15 is that using the global similarity transform is the better option, at least for this video sequence. The frequency of convergence results in Figure 15 are consistently better than the corresponding results in Figure 14.

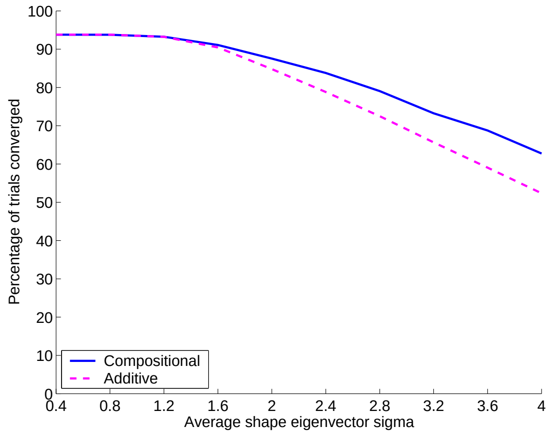
5.4 Experiment 2: Computation of the Steepest Descent Images

The inverse compositional algorithm uses the *steepest descent images* $\boldsymbol{\nabla} A_0 \frac{\partial \mathbf{W}}{\partial \mathbf{p}}$. See [Baker and Matthews, 2003] for a description of why these images can be called the *steepest descent images.* The analytically derived steepest descent images play exactly the same role as the numerically estimated images $\frac{\partial \mathbf{r}}{\partial \mathbf{p}}$ in [Cootes *et al.*, 2001, Cootes, 2001] (and the corresponding finite difference images in [Sclaroff and Isidoro, 1998, Gleicher, 1997].) The difference between these two sets of images is that in the additive formulation, the images $\frac{\partial \mathbf{r}}{\partial \mathbf{p}}$ are a function of $\mathbf{p}$ whereas in the inverse compositional formulation they are provably constant. Otherwise their roles are identical.

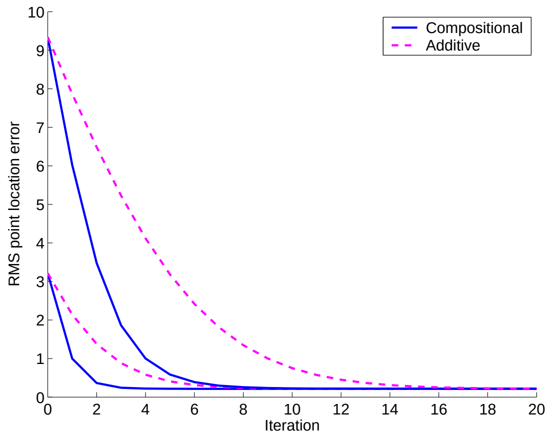
Is it better to analytically estimate $\boldsymbol{\nabla} A_0 \frac{\partial \mathbf{W}}{\partial \mathbf{p}}$ or to numerically estimate $\frac{\partial \mathbf{r}}{\partial \mathbf{p}}$? To answer this question, we compare the inverse compositional algorithm with the two different estimates, the analytical estimate described above, and the numerical estimate of $\frac{\partial \mathbf{r}}{\partial \mathbf{p}}$ as described in [Cootes *et al.*, 2001]. (All other aspects of the algorithm are exactly the same. Specifically we use the inverse compositional update in both cases.) The results in Figure 16 show that the analytically derived steepest descent images performing significantly better. The reason for this difference is probably a combination of: (1) the computation of the analytically derived steepest descent images does not depend on the background and (2) the computation of the analytically derived steepest descent images is performed at $\mathbf{p} = \mathbf{0}$ whereas the numerical equivalents are an average over $\mathbf{p}$.



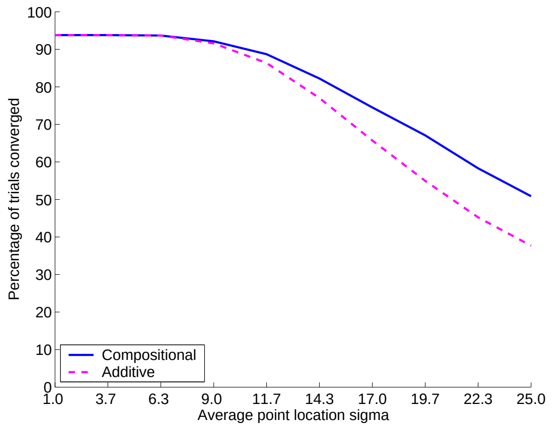
(a) Rate of Convergence Perturbing Shape



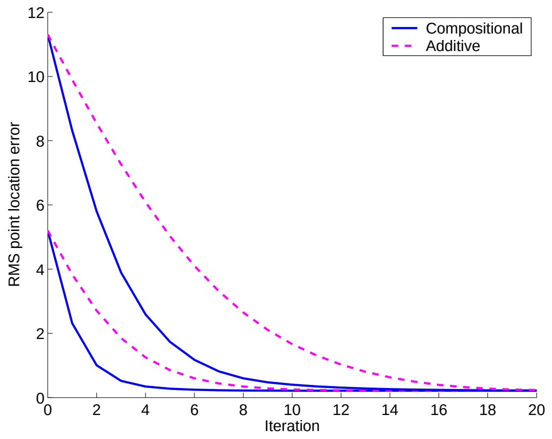
(b) Frequency of Convergence Perturbing Shape



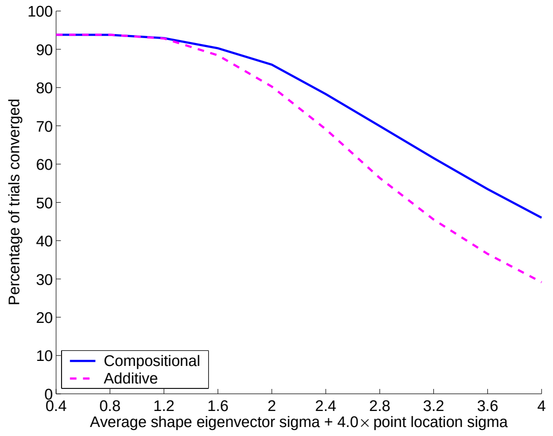
(c) Rate of Convergence Perturbing Similarity



(d) Frequency of Convergence Perturbing Similarity

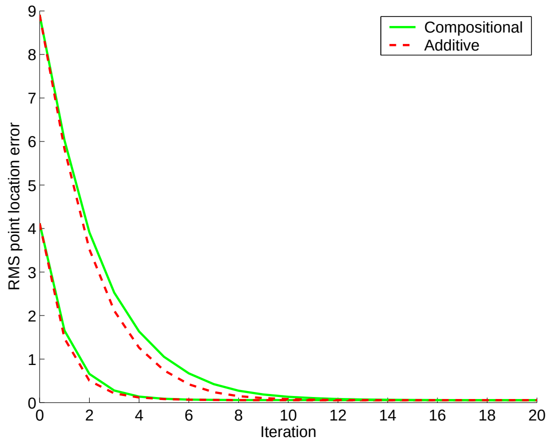


(e) Rate of Convergence Perturbing Both

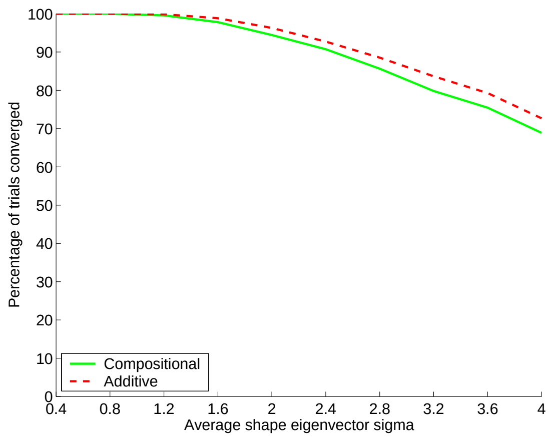


(f) Frequency of Convergence Perturbing Both

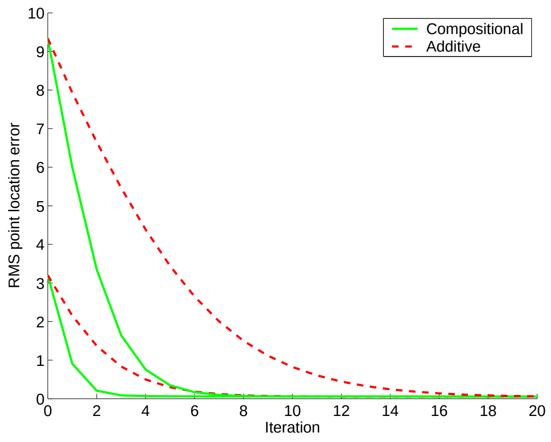
Figure 14: The results of comparing the additive and compositional updates to the warp in Step 9 of the algorithm without including a global shape normalizing transform. We plot the rate of convergence in (a), (c), and (e), and the frequency of convergence in (b), (d), and (f). In (a) and (b) we just perturb the shape parameters, in (c) and (d) we just perturb the similarity parameters, and in (e) and (f) we perturb both sets of parameters. The results in all cases show the compositional update to be better than the additive update.



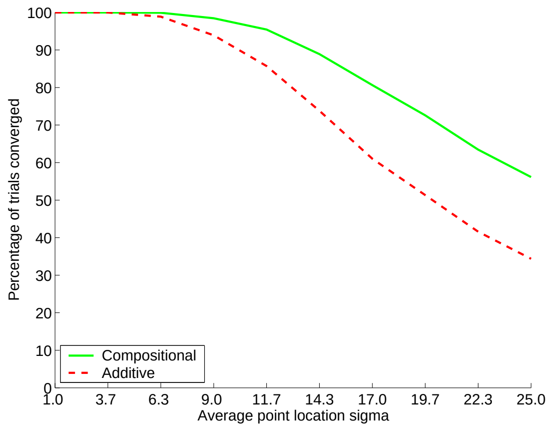
(a) Rate of Convergence Perturbing Shape



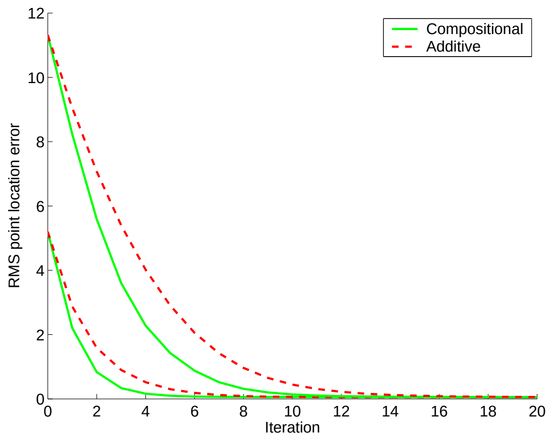
(b) Frequency of Convergence Perturbing Shape



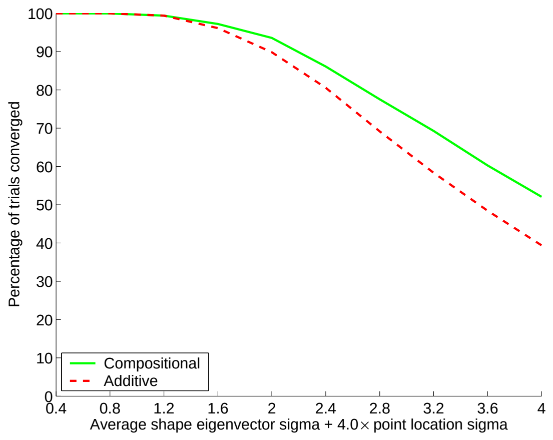
(c) Rate of Convergence Perturbing Similarity



(d) Frequency of Convergence Perturbing Similarity

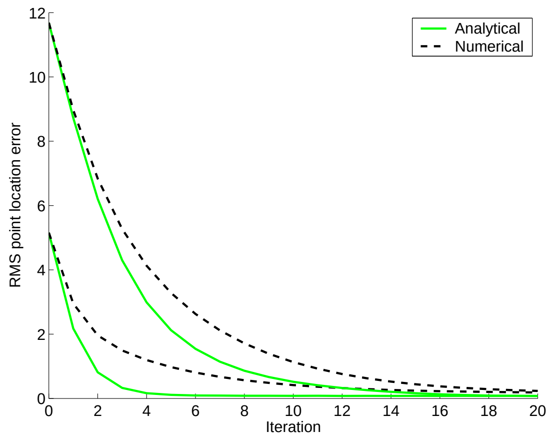


(e) Rate of Convergence Perturbing Both

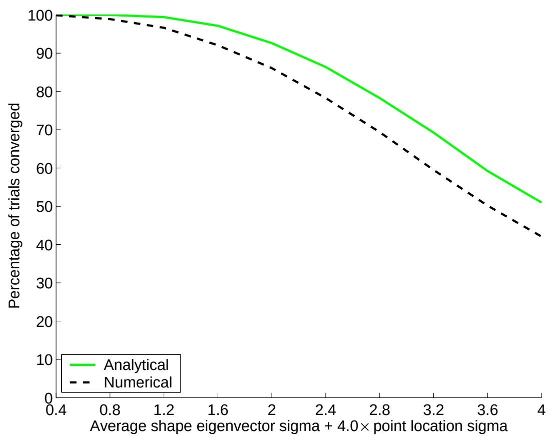


(f) Frequency of Convergence Perturbing Both

Figure 15: The results of comparing the additive and compositional updates to the warp in Step 9 of the algorithm including a global shape normalizing transform (a similarity transform.) We perform the same experiments as in Figure 14. These results with the global similarity transform are all generally better than the corresponding results in Figure 14. The compositional update generally outperforms the additive update. The one exception is in (a) and (b). The global similarity transform partially hides the problem described in Section 2.3.3, however the problem is just transfered to the similarity parameters and so the overall results in (e) and (f) still show that the additive update performs worse than for the compositional update.



(a) Rate of Convergence Perturbing Both



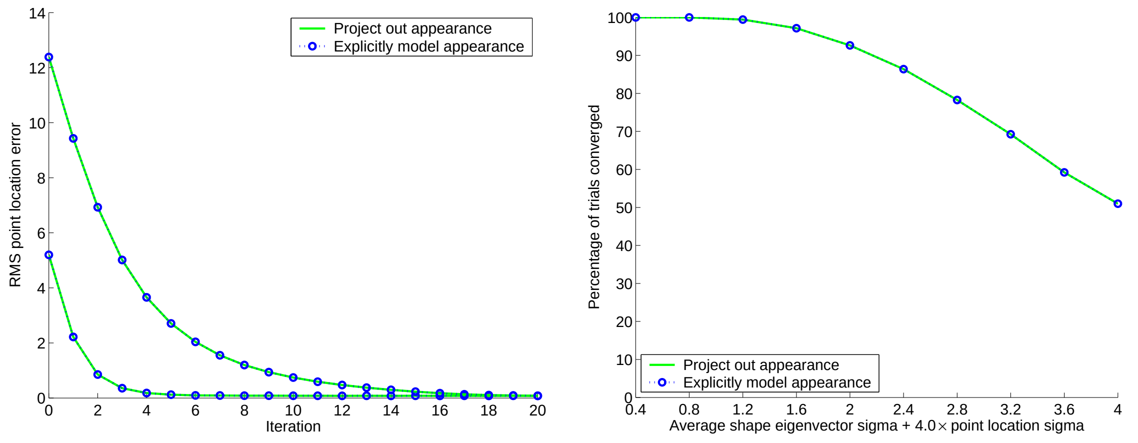
(b) Frequency of Convergence Perturbing Both

Figure 16: A comparison of using the analytically derived steepest descent images $\nabla A_0 \frac{\partial \mathbf{W}}{\partial \mathbf{p}}$ versus the numerical finite difference images $\frac{\partial \mathbf{r}}{\partial \mathbf{p}}$ used in [Cootes *et al.*, 2001, Cootes, 2001, Sclaroff and Isidoro, 1998, Gleicher, 1997]. The results show that the analytically derived images are significantly better.

5.5 Experiment 3: Appearance Variation

In our algorithm we "project out" the appearance variation. See Section 4.2. Projecting out the appearance has the advantage that there are less unknowns and so the algorithm is more efficient. This approach should be contrasted with that in [Cootes *et al.*, 2001, Cootes, 2001, Sclaroff and Isidoro, 1998, Gleicher, 1997] where the appearance parameters are treated like any other and are iteratively updated. There are potentially benefits of iteratively updating the appearance parameters. In particular, any correlation between the appearance and shape parts of the model can be accounted for in the Hessian matrix. It is straightforward to modify the inverse compositional algorithm to model the appearance variation in this way. We just need to compute steepest descent images for the appearance part of the model which is easy. The steepest descent image for the appearance parameter λ_i is $A_i(\mathbf{x})$. These steepest descent images are simply appended to the vector of steepest descent images, the Hessian increases in size appropriately, and otherwise the algorithm remains the same. The additive approach is the correct way to update the appearance parameters.

We plot results comparing the "project out appearance" approach with the "explicitly model appearance" approach in Figure 17. Again, all other aspects of the two algorithms are identical. Both use the inverse compositional update and the analytical steepest descent images. The results in Figure 17 show that the two approaches perform almost identically. Note, however, that the project out approach is far more efficient because less parameters are updated in each iteration of the algorithm. The "project out appearance" approach is therefore the better choice.



(a) Rate of Convergence Perturbing Both (b) Frequency of Convergence Perturbing Both

Figure 17: A comparison of the "project out appearance" approach described in Section 4.2 with the usual approach of "explicitly modeling appearance" used in [Cootes *et al.*, 2001, Cootes, 2001]. The results show no significant difference in the performance of the two approaches. The "project out appearance" approach is therefore the better choice because it is far more computationally efficient. See Table 1.

5.6 Computational Efficiency

One concern with the inverse compositional algorithm is that the time taken to perform the inverse compositional update might be quite long. Although the description of the update is quite involved, the actual calculation is minimal because the number of vertices in the mesh is far less than the number of pixels. In Table 1 we include timing results for our Matlab implementations of the four algorithms compared in this section. The results were obtained on a dual 2.4GHz P4 machine and are for a model with 19,977 pixels, 3 shape parameters, 4 similarity parameters, and 9 appearance parameters. As can be seen, the inverse compositional update in Step 9 is negligible compared with the other steps and so the algorithm is only a little slower than the additive algorithm. Also note: (1) that the analytical computation of the steepest descent images in Steps 3, 4, and 5 is significantly faster than the numerical computation and (2) the explicitly modeling appearance algorithm is significantly slower both because the computation of the error image in Step 8 is more involved and because there are more parameters to solve for in each iteration. Overall our Matlab implementation of the inverse compositional algorithm operates at approximately 5Hz. Our "C" implementation operates at approximately 150Hz on the same machine.

6 Conclusion

6.1 Summary

Previous AAM fitting algorithm fall into one of two categories: (1) inefficient gradient descent algorithms and (2) efficient, but ad-hoc, algorithms that make the invalid assumption that there is a constant linear relationship between the error image and the additive updates to the (shape)

Table 1: Timing results for our Matlab implementations of the four algorithms evaluated in Section 5 in milliseconds. These results were obtained on a dual 2.4GHz P4 machine and are for a model with 19,977 pixels, 3 shape parameters, 4 similarity parameters, and 9 appearance parameters. The main things to note are: (1) the inverse compositional algorithm is not significantly slower than the additive algorithm, (2) the pre-computation of the numerical steepest descent images is slow, and (3) explicitly modeling appearance is significantly slower than projecting out appearance. Overall the inverse compositional algorithm runs at about 5Hz in Matlab. The "C" implementation runs at approximately 150Hz on the same machine.

Pre-computation:

	Step 3	Step 4	Step 5	Step 6	Total
Inverse Compositional	12,682	5,726	90.2	85.2	18,583
With Additive Update	12,682	5,726	90.2	85.2	18,583
With Numerical Gradient	-	-	251,012	85.2	251,097
Explicitly Modeling Appearance	12,682	5,726	34.3	435.5	18,878

Per Iteration:

	Step 1	Step 2	Step 7	Step 8	Step 9	Total
Inverse Compositional	2.7	3.0	12.0	0.1	0.2	17.8
With Additive Update	2.7	3.0	12.0	0.1	0.1	17.7
With Numerical Gradient	2.7	3.0	12.0	0.1	0.2	17.8
Explicitly Modeling Appearance	2.7	6.4	26.7	0.1	0.3	36.2

Post Computation:

	Step 10
Inverse Compositional	7.2
With Additive Update	7.2
With Numerical Gradient	7.2
Explicitly Modeling Appearance	-

parameters. In this paper we have proposed an algorithm for fitting AAMs that has the advantages of both types of algorithms. It is an analytical gradient descent algorithm with well understood convergence properties that is even more efficient than the ad-hoc algorithms. The algorithm is an extension to our inverse compositional image alignment algorithm [Baker and Matthews, 2003]. Overall our algorithm outperforms previous approaches in terms of: (1) the speed of convergence (far fewer iterations are needed to converge to any give accuracy), (2) the frequency of convergence (our algorithm is more likely to convergence from a large distance away), and (3) the computational cost (the algorithm is far faster because the appearance variation is projected out.)

6.2 Discussion

The inverse compositional AAM fitting algorithm can only be applied to *independent* AAMs. It cannot be applied to *combined* AAMs which parameterize the shape and appearance variation with a single set of parameters and so introduce a coupling between shape and appearance. Although this may seem like a serious limitation because combined AAMs can parameterize the same visual phenomenon with fewer parameters, in practice it is not. The nonlinear optimization in our algorithm is only over the n shape parameters and so is actually lower dimensional than the equivalent combined AAM optimization which would have more than n parameters. Currently we do not see a way to extend our algorithm to combined AAMs, but of course we may be wrong.

6.3 Future Work

The inverse compositional algorithm is an image alignment algorithm. Hence most standard extensions to image alignment can be applied with this algorithm. See Section 4.4. For other extensions, such as the use of robust norms [Sclaroff and Isidoro, 1998, Hager and Belhumeur, 1998, Black and Jepson, 1998], the situation is not quite to clear cut. For example, when a robust norm is used the orthogonal decomposition of the norm in Equation (39) is not applicable. The efficient treatment of appearance is therefore more complex. Similarly, the introduction of priors on the shape and appearance parameters is not quite so easy with the inverse compositional algorithm. In future work we hope to look into these questions and determine which of these extensions can be used with our inverse compositional AAM fitting algorithm and which cannot.

Acknowledgments

We thank Tim Cootes for discussions on the incorporation of the global shape transformation in Section 4.3. Elements of the AAM fitting algorithm appeared in [Baker and Matthews, 2001]. We thank the reviewers of [Baker and Matthews, 2001] for their feedback. The research described in this paper was conducted under U.S. Office of Naval Research contract N00014-00-1-0915. Additional support was provided through U.S. Department of Defense award N41756-03-C4024.

References

[Baker and Matthews, 2001] S. Baker and I. Matthews. Equivalence and efficiency of image alignment algorithms. In *Proceedings of CVPR*, volume 1, pages 1090–1097, 2001.

[Baker and Matthews, 2003] S. Baker and I. Matthews. Lucas-Kanade 20 years on: A unifying framework: Part 1: The quantity approximated, the warp update rule, and the gradient descent approximation. *Accepted to Appear in the International Journal of Computer Vision*, 2003. Previously appeared as CMU Robotics Institute Technical Report CMU-RI-TR-02-16.

[Bergen *et al.*, 1992] J.R. Bergen, P. Anandan, K.J. Hanna, and R. Hingorani. Hierarchical model-based motion estimation. In *Proceedings of ECCV*, pages 237–252, 1992.

[Black and Jepson, 1998] M. Black and A. Jepson. Eigen-tracking: Robust matching and tracking of articulated objects using a view-based representation. *IJCV*, 36(2):101–130, 1998.

[Blanz and Vetter, 1999] 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.

[Cootes and Kittipanya-ngam, 2002] T.F. Cootes and P. Kittipanya-ngam. Comparing variations on the active appearance model algorithm. In *Proc. of BMVC*, volume 2, pages 837–846, 2002.

[Cootes *et al.*, 1998] T.F. Cootes, G.J. Edwards, and C.J. Taylor. Active appearance models. In *Proceedings of the European Conference on Computer Vision*, volume 2, pages 484–498, 1998.

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

[Cootes, 2001] T. F. Cootes. Statistical models of appearance for computer vision. Online technical report available from http://www.isbe.man.ac.uk/˜bim/refs.html, 2001.

[Gleicher, 1997] M. Gleicher. Projective registration with difference decomposition. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 331–337, 1997.

[Hager and Belhumeur, 1998] G.D. Hager and P.N. Belhumeur. Efficient region tracking with parametric models of geometry and illumination. *PAMI*, 20(10):1025–1039, 1998.

[Hou *et al.*, 2001] X. Hou, S. Li, H. Zhang, and Q. Cheng. Direct appearance models. In *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition*, volume 1, pages 828–833, 2001.

[Jones and Poggio, 1998] M. Jones and T. Poggio. Multidimensional morphable models: A framework for representing and matching object classes. In *Proc. ICCV*, pages 683–688, 1998.

[Lanitis *et al.*, 1997] A. Lanitis, C. J. Taylor, and T. F. Cootes. Automatic interpretation and coding of face images using flexible models. *PAMI*, 19(7):742–756, 1997.

[Lucas and Kanade, 1981] B. Lucas and T. Kanade. An iterative image registration technique with an application to stereo vision. In *Proc. of the IJCAI*, pages 674–679, 1981.

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

[Shum and Szeliski, 2000] H.-Y. Shum and R. Szeliski. Construction of panoramic image mosaics with global and local alignment. *International Journal of Computer Vision*, 16(1):63–84, 2000.

[Vetter and Poggio, 1997] T. Vetter and T. Poggio. Linear object classes and image synthesis from a single example image. *IEEE Transactions on PAMI*, 19(7):733–742, 1997.