Adaptive and Constrained Algorithms for Inverse Compositional Active Appearance Model Fitting

George Papandreou and Petros Maragos
School of E.C.E., National Technical University of Athens, Greece
http://cvsp.cs.ntua.gr*

Abstract

Parametric models of shape and texture such as Active Appearance Models (AAMs) are diverse tools for deformable object appearance modeling and have found important applications in both image synthesis and analysis problems. Among the numerous algorithms that have been proposed for AAM fitting, those based on the inverse-compositional image alignment technique have recently received considerable attention due to their potential for high efficiency. However, existing fitting algorithms perform poorly when used in conjunction with models exhibiting significant appearance variation, such as AAMs trained on multiple-subject human face images. We introduce two enhancements to inverse-compositional AAM matching algorithms in order to overcome this limitation. First, we propose fitting algorithm ***adaptation****, by means of (a) fitting matrix adjustment and (b) AAM mean template update. Second, we show how prior information can be incorporated and* ***constrain*** *the AAM fitting process. The inverse-compositional nature of the algorithm allows efficient implementation of these enhancements. Both techniques substantially improve AAM fitting performance, as demonstrated with experiments on publicly available multi-person face datasets.*

1. Introduction

Parametric models of shape and texture, such as Active Appearance Models [8], Active Blobs [25], Morphable Models [18], and other related approaches [5, 11, 16] are widely used techniques for object appearance modeling. Employing a number of parameters controlling shape and texture variation, these models bring a target image into registration with a reference template, even in cases that the target image is a deformed version of the template; imaging conditions such as camera position and object illumination can also differ significantly between the template and the target image. Active Appearance Models can additionaly represent appearance variability in a whole class of objects, such as faces or cars, after learning it from examples during a training phase. Such parametric models can be applied to both image synthesis and analysis problems. Representative applications are object tracking in video [25], face recognition [6], face synthesis [18], and image stitching [26].

An important issue with AAMs concerns matching them to images by finding the parameters that minimize the discrepancy between observed and synthesized object appearances, possibly also including a prior penalty on the parameter values. This is a difficult non-linear optimization task and general-purpose optimization procedures can be inneficient. Most existing AAM fitting algorithms solve the matching problem by iteratively updating the model parameters, assuming that there is a fixed *AAM fitting matrix* which maps the synthesis error image to model parameter increments; this matrix is learned in a precomputation phase and is subsequently used unaltered, resulting in a very efficient class of algorithms, reviewed in Section 2. However, the fixed mapping approach reaches its limits when one works with models allowing considerable appearance deviation from the mean template, such as AAMs built on large multi-person face datasets [3, 10].

As has been highlighted in [2], in performing incremental image registration one can choose between (a) updating the warp parameters either additively or compositionally and (b) calculating incremental warps and image gradients either forwardly on the target image or inversely on the model template. In our work, similarly to [2, 12, 22], we use the *inverse compositional* combination because it facilitates efficient calculations, since manipulating the model template instead of the target image allows precomputing useful quantities and thus accelerates image fitting.

Our first main contribution is that we introduce two complementary **adaptation** mechanisms in inverse compositional AAM fitting. The first adaptation mechanism amounts to *AAM fitting matrix adjustment* and compensates

*This work was supported in part by grant ΠΕΝΕΔ-2003-ΕΔ865 [co-financed by E.U.-European Social Fund (75%) and the Greek Ministry of Development-GSRT (25%)] and in part by the European Community projects ASPI, MUSCLE, and HIWIRE.

978-1-4244-2243-2/08/$25.00 ©2008 IEEE

for the deviation of the model's synthetic texture from the mean texture. A desirable characteristic of our algorithm is that it features a tunable order parameter which adjusts the adaptation accuracy *vs*. computational load trade-off, allowing a variety of choices to the user. We show that previously presented algorithms such as the fast (but possibly inaccurate) project-out method of [19] and the more accurate (but slow) simultaneous method of [15] are just the zero-order and maximum-order, respectively, extremes of plausible choices. Our analysis sheds new light in their properties and their shortcomings. As a second complementary adaptation mechanism, we propose a computationally cheap *mean template update* procedure, particularly suited for object tracking in videos, which can be applied periodically and adapt the mean texture of the model's template, considerably improving the accuracy of the fast (low-order) algorithms we present. Both adaptation techniques are presented in Section 3.

Our second main contribution is that we show how **prior constraints** on model parameters can be properly incorporated into inverse compositional AAM fitting algorithms. Such prior information, typically in the form of dynamic constraints within a tracking system or static constraints induced during the PCA-based AAM training phase, can significantly improve AAM fitting robustness as has been demonstrated within the conventional forwards additive parameter update framework [9]. However, utilizing them within the compositional warp update mechanism is not straightforward, as one needs to compute the Jacobian matrix of the compositional-to-additive warp update, which had only been done before for the simple case of global affine transformation [1]. We show in Section 4 how this Jacobian matrix can be efficiently calculated for the more flexible warps often utilized in conjunction with AAMs, such as thin-plate spline warps. We present in Section 5 experimental results on face matching and tracking with AAMs built on multi-person datasets which demonstrate that incorporating adaptation mechanisms and prior constraints substantially improves AAM fitting performance.

2. Active Appearance Models

Active Appearance Models are generative models which use a compact set of parameters to describe the shape and texture variation of objects in images.

2.1. Object Appearance Representation

Typically the shape of the object is sampled at L landmarks, whose coordinates constitute a shape vector $\mathbf{s}$ of length $2L$ in the 2-D case. Active Appearance Models allow a particular instance of the shape $\mathbf{s_p}$ deviate from a mean shape $\mathbf{s}_0$ by letting $\mathbf{s_p} - \mathbf{s}_0$ lie in a linear subspace spanned by n eigenshapes $\mathbf{s}_i$, yielding

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

The modes of shape variation $\mathbf{s}_i$ can be either statistically learned using a training set [8], or computed by modal analysis of the shape mesh [25], or selected a-priori to allow modeling certain distortions [14]. Often these modes deliberately do not model scale and translation, in which case an explicit 4 d.o.f. similarity transform $\mathbf{S_t}$, defined as $\mathbf{S_t}(\mathbf{x}) = \left(\begin{smallmatrix} 1+t_1 & -t_2 \\ t_2 & 1+t_1 \end{smallmatrix}\right)\mathbf{x} + \left(\begin{smallmatrix} t_3 \\ t_4 \end{smallmatrix}\right)$, makes the model scale and translation invariant. The enhanced shape parameter vector $\tilde{\mathbf{p}} = [\mathbf{t}_{1:4}, \mathbf{p}_{1:n}]^T$ with length $4+n$ implicitly defines a dense continuous deformation field $\mathbf{W}(\mathbf{x}, \tilde{\mathbf{p}}) = \mathbf{S}_t\big(\mathbf{W}(\mathbf{x}, \mathbf{p})\big)$, namely deformation followed by similarity, which maps every point $\mathbf{x}$ in the model template to its corresponding image point as follows: The deformation $\mathbf{W}(\mathbf{x}, \mathbf{p})$, typically a thin-plate spline or a piecewise affine mapping, is determined by requiring that it maps each landmark in the reference shape $\mathbf{s}_0$ to its corresponding landmark in $\mathbf{s_p}$.

The texture part of the appearance refers to the intensity or color (other information channels can also be added) of the object in a shape-normalized frame, after registering it with the model template. Similarly to shape, allowable texture samples $A_{\boldsymbol{\lambda}}(\mathbf{x})$ are generated linearly, using a mean texture $A_0(\mathbf{x})$ and a set of m eigentextures $A_i(\mathbf{x})$:

$$A_{\boldsymbol{\lambda}} = A_0 + \sum_{i=1}^{m} \lambda_i A_i, \tag{2}$$

where we have used vector notation for textures; *e.g.* A_0 denotes the mean texture image raster-scanned into a vector with N entries, as many as the texture samples of the reference object. The eigentexture images compensate illumination changes [16] and model texture variability between different objects of the same class (*e.g.* faces) [8, 18]. For example, Figure 1 shows the leading eigenshapes and eigentextures obtained by AAM training on a person's face. Camera gain and offset are usually accounted for separately by a global affine texture transformation $T_{\mathbf{u}}(I) = (u_1+1)I + u_2$. We gather all texture parameters in an enhanced texture vector $\tilde{\boldsymbol{\lambda}} = [\mathbf{u}_{1:2}, \boldsymbol{\lambda}_{1:m}]^T$ with length $2+m$.

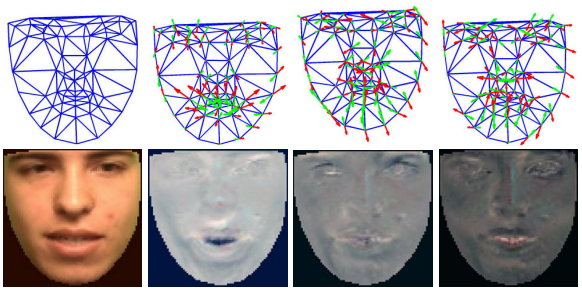

Figure 1. *Upper row*: Mean shape $\mathbf{s}_0$ and the first eigenshapes $\mathbf{s}_i$. *Bottom row*: Mean texture A_0 and the first eigentextures A_i.

2.2. Model Fitting

A central issue with parametric appearance models is designing algorithms that efficiently and accurately fit them to a novel target image I, *i.e*. find the concatenated shape and texture parameter vector $\mathbf{q} = [\tilde{\mathbf{p}}^T, \tilde{\boldsymbol{\lambda}}^T]^T$ with length $n + m + 6$ that minimizes the discrepancy between the warped-back normalized image texture $T_{\mathbf{u}}(I(\mathbf{W}(\tilde{\mathbf{p}})))$ and the synthesized texture $A_{\boldsymbol{\lambda}}$. The error image $E(\mathbf{q})$ is:

$$\begin{aligned} E(\mathbf{q}) &= T_{\mathbf{u}}(I(\mathbf{W}(\tilde{\mathbf{p}}))) - A_{\boldsymbol{\lambda}} \\ &= T_{\mathbf{u}}(I(\mathbf{W}(\tilde{\mathbf{p}}))) - (A_0 + \sum_{i=1}^{m} \lambda_i A_i). \end{aligned} \tag{3}$$

The mismatch is usually quantified by the Euclidean norm $\frac{1}{\sigma^2}\|E(\mathbf{q})\|_2^2$ (sum of square differences) of the error image, where σ^2 is the variance of the model noise; robust norms are advantageous when handling occlusion [5]. Minimizing this mis-match is a non-linear least-squares problem on a high-dimensional space and general-purpose optimization techniques such as stochastic gradient descent [18] can be slow. Most efficient techniques to solve the problem require as input a good starting guess for the unknown parameter vector $\mathbf{q}$ and then iteratively update it until a (local) minimum of the mismatch norm is reached. Building an image pyramid and matching in a coarse-to-fine fashion typically improves the robustness of such incremental methods [4].

A standard general technique for improving the parameter estimate uses a first-order Taylor expansion $E(\mathbf{q} + d\mathbf{q}) \approx E(\mathbf{q}) + \frac{\partial E}{\partial \mathbf{q}} d\mathbf{q}$ and then applies a Gauss-Newton type algorithm to compute an additive increment by $d\mathbf{q} = K(\mathbf{q})E(\mathbf{q})$, where $K(\mathbf{q}) = -(\frac{\partial E}{\partial \mathbf{q}}^T \frac{\partial E}{\partial \mathbf{q}})^{-1} \frac{\partial E}{\partial \mathbf{q}}^T$. However this is computationally expensive, since image gradients $\frac{\partial I}{\partial \mathbf{x}}$ and warp Jacobians $\frac{\partial \mathbf{W}}{\partial \mathbf{p}}$ need to be recomputed at every step [2]. Although $K(\mathbf{q})$ changes in general, a number of authors still utilize a fixed *AAM fitting matrix* K from $E(\mathbf{q})$ to $d\mathbf{q}$ which is computed by multivariate analysis on the training set [8, 13, 14, 25]. An interesting recent extension of this approach allows the fixed mapping to be non-linear, and learns it on the training set via boosting [23]. Utilizing a fixed linear or non-linear mapping to compute parameter updates leads to very efficient algorithms which often demonstrate good accuracy. However, it has been demonstrated that adapting the mapping $K(\mathbf{q})$ to the target image I can lead to notable performance improvements, especially in dealing with target images whose texture substantially departs from the mean model texture [3, 10].

As Baker and Matthews have highlighted [2, 19], the so-called *forwards additive* class of algorithms just described is not the only viable parameter update strategy. They unified previous work on *forwards compositional* [24] and *inverse additive* [16] parameter update strategies in iterative image alignment algorithms and introduced the *inverse compositional* parameter update technique, where a warp parameter update $d\tilde{\mathbf{p}}$ is combined with the current estimate $\tilde{\mathbf{p}}$ inverse-compositionally:

$$\mathbf{W}(\mathbf{x}, \tilde{\mathbf{p}}) \leftarrow \mathbf{W}(\mathbf{x}, \tilde{\mathbf{p}}) \circ \mathbf{W}^{-1}(\mathbf{x}, d\tilde{\mathbf{p}}) \equiv \mathbf{W}(\mathbf{W}^{-1}(\mathbf{x}, d\tilde{\mathbf{p}}), \tilde{\mathbf{p}}). \tag{4}$$

They showed that, although the compositional parameter update (4) is obviously more costly than the simple additive update $\tilde{\mathbf{p}} \leftarrow \tilde{\mathbf{p}} + d\tilde{\mathbf{p}}$, each full step of the IC algorithm is overall cheaper than in any alternative approach when texture variation is allowed, because it turns out that most of the quantities involved do not change during the fitting procedure and thus can be precomputed, as will be made clear in the sequel. See [2] for further details and [22] for an application of the inverse compositional approach to 3-D morphable model fitting.

Exploiting its advantageous properties, Baker *et al*. have introduced two algorithms that fall into the inverse compositional framework. On the one hand, their *project-out* algorithm [19] avoids updates to the texture parameters $\boldsymbol{\lambda}$ and matrix inversions every step and is thus extremely efficient; however it performs very poorly when the texture variability in the object class is big and the authors have not precisely identified the reasons of this failure [15]. On the other hand, their *simultaneous* algorithm [15] is more accurate but also fairly slow. The adaptation mechanisms which we introduce in Section 3 shed new light and move beyond the project-out and simultaneous algorithms. Moreover, including prior information in the fitting process as described in Section 4 clearly improves further the accuracy and robustness of the inverse compositional family of algorithms.

3. Adaptive Inverse Compositional AAMs

In this section we discuss two adaptation strategies for Inverse Compositional AAMs. The first is an efficient variable-order algorithm which adjusts the AAM fitting matrix, adapting to the target image texture. The second is a mean template update strategy which periodically modifies the mean texture vector and significantly improves the accuracy of our low-order inverse compositional algorithms.

3.1. Fitting Matrix Adjustment

We work in the inverse compositional framework reviewed in Section 2 and use similar notation to [15, 19]. In each fitting step we look for a parameter update vector $d\mathbf{q} = [d\tilde{\mathbf{p}}^T, d\tilde{\boldsymbol{\lambda}}^T]^T$ that minimizes the norm $\|E(\mathbf{q}, d\mathbf{q})\|_2$ of the error image with respect to $d\mathbf{q}$. From Eq. (3), the error at an arbitrary point $\mathbf{x}$ in the reference patch after the inverse-compositional update is applied is given by

$$\begin{aligned} E(\mathbf{x}; \mathbf{q}, d\mathbf{q}) = T_{\mathbf{u}}\big(I(\mathbf{W}(\mathbf{x}; \tilde{\mathbf{p}}))\big) - \\ T_{d\mathbf{u}}\big(A_{\boldsymbol{\lambda}+d\boldsymbol{\lambda}}(\mathbf{W}(\mathbf{x}; d\tilde{\mathbf{p}}))\big) \end{aligned} \tag{5}$$

Making a first-order Taylor expansion around zero $d\mathbf{q}$, ignoring second-order terms and applying the chain-rule to compute derivatives of composite functions, yields:

$$E(\mathbf{x};\mathbf{q},d\mathbf{q}) = E(\mathbf{x};\mathbf{q}) - \frac{\partial T(A_{\boldsymbol{\lambda}}(\mathbf{x}))}{\partial \mathbf{u}} d\mathbf{u} - \frac{\partial A_{\boldsymbol{\lambda}}(\mathbf{x})}{\partial \mathbf{x}} \frac{\partial \mathbf{W}(\mathbf{x})}{\partial \tilde{\mathbf{p}}}\Big|_{\tilde{\mathbf{p}}=\mathbf{0}} d\tilde{\mathbf{p}} - [A_1(\mathbf{x}) \dots A_m(\mathbf{x})] d\boldsymbol{\lambda}. \quad (6)$$

The images multiplying the parameter updates, called *steepest descent* (s.d.) images in [19], give the change in texture caused by updating the corresponding parameter. These steepest descent images in our case are: 1) brightness correction s.d. images $\frac{\partial T(A_{\boldsymbol{\lambda}}(\mathbf{x}))}{\partial \mathbf{u}} = [A_{\boldsymbol{\lambda}}(\mathbf{x})\, 1]$ corresponding to $d\mathbf{u}$, 2) texture variation s.d. images $[A_1(\mathbf{x}) \dots A_m(\mathbf{x})]$ corresponding to $d\boldsymbol{\lambda}$, and 3) shape warp s.d. images, also called *motion templates* in [16], $M_{\boldsymbol{\lambda}}(\mathbf{x}) = \frac{\partial A_{\boldsymbol{\lambda}}(\mathbf{x})}{\partial \mathbf{x}} \frac{\partial \mathbf{W}(\mathbf{x})}{\partial \tilde{\mathbf{p}}}$, corresponding to the $2+n$ shape parameters in $d\tilde{\mathbf{p}}$.

We make two observations. First, by reparameterizing the texture increment vector as

$$d\boldsymbol{\lambda}' = d\boldsymbol{\lambda} - u_1 \boldsymbol{\lambda} \quad (7)$$

and defining its corresponding enhanced version $d\tilde{\boldsymbol{\lambda}}' = [d\mathbf{u}_{1:2}, d\boldsymbol{\lambda}'_{1:m}]^T$, yields the image-independent merged brightness correction and texture variation s.d. images $A(\mathbf{x}) = [A_0(\mathbf{x})\, 1\, A_1(\mathbf{x}) \dots A_m(\mathbf{x})]$ corresponding to $d\tilde{\boldsymbol{\lambda}}'$. Second, note that the warp Jacobian $\frac{\partial \mathbf{W}(\mathbf{x})}{\partial \tilde{\mathbf{p}}} = \left[\frac{\partial \mathbf{S}}{\partial \mathbf{t}}, \frac{\partial \mathbf{W}(\mathbf{x})}{\partial \mathbf{p}}\right]$ is evaluated for zero $\tilde{\mathbf{p}}$; thus, as expected from the inverse compositional strategy we have adopted, it remains fixed. However the motion templates $M_{\boldsymbol{\lambda}}(\mathbf{x})$ do depend on the current texture estimate $\boldsymbol{\lambda}$ through the template gradient term $\frac{\partial A_{\boldsymbol{\lambda}}(\mathbf{x})}{\partial \mathbf{x}}$ and can be written as:

$$M_{\boldsymbol{\lambda}} = M_0 + \sum_{i=1}^{m} \lambda_i M_i, \quad (8)$$

where $M_i(\mathbf{x}) = \frac{\partial A_i(\mathbf{x})}{\partial \mathbf{x}} \frac{\partial \mathbf{W}(\mathbf{x})}{\partial \tilde{\mathbf{p}}}\Big|_{\tilde{\mathbf{p}}=\mathbf{0}}$. Here M_0 gathers the unadapted motion templates corresponding to the model's mean texture A_0, while each M_i, for $i = 1, \dots, m$, is a *fixed* $N \times (n+4)$ matrix isolating the contribution of the A_i eigen-texture to the motion templates. The effect of such an adaptation on the motion template corresponding to x-translation (parameter t_3) is illustrated on Fig. 2. Adapting the motion templates with Eq. (8) thus compensates for texture variation within the inverse compositional framework.

Resorting back to vector notation for images, we can rewrite compactly the least squares problem to be solved as

$$\min \frac{1}{2\sigma^2} \|E(\mathbf{q}) - H_{\boldsymbol{\lambda}} d\mathbf{q}'\|_2^2, \quad (9)$$

where $E(\mathbf{q})$ is the current texture error given by Eq. (3); $H_{\boldsymbol{\lambda}} = \begin{bmatrix} M_{\boldsymbol{\lambda}} & A \end{bmatrix}$ is the $N \times (m+n+6)$ compound matrix of

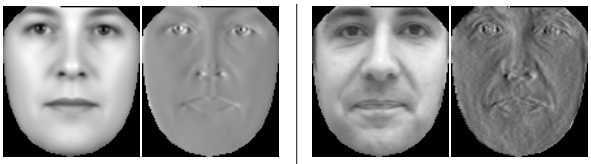

Figure 2. *Left*: Mean texture A_0 and x-tranlation mean motion template (third column of M_0). *Right*: Target image I and corresponding adapted motion template (third column of $M_{\boldsymbol{\lambda}}$).

steepest descent images; and $d\mathbf{q}' = \begin{bmatrix} d\tilde{\mathbf{p}} \\ d\tilde{\boldsymbol{\lambda}}' \end{bmatrix}$ is the vector of the shape and reparameterized texture parameters. Invoking the normal equations yields, respectively, the least-squares estimate and corresponding covariance matrix

$$d\mathbf{q}' = \sigma^{-2} \Sigma_{\mathbf{q}} H_{\boldsymbol{\lambda}}^T E(\mathbf{q}) \quad (10a)$$

$$\Sigma_{\mathbf{q}} = \sigma^2 (H_{\boldsymbol{\lambda}}^T H_{\boldsymbol{\lambda}})^{-1} = \sigma^2 \begin{bmatrix} M_{\boldsymbol{\lambda}}^T M_{\boldsymbol{\lambda}} & M_{\boldsymbol{\lambda}}^T A \\ A^T M_{\boldsymbol{\lambda}} & A^T A \end{bmatrix}^{-1}. \quad (10b)$$

Similarly to [16], we can consider Eq. (10) in partitioned form, separating the shape and texture components of the solution. This yields the shape parameter update

$$d\tilde{\mathbf{p}} = (M_{\boldsymbol{\lambda}}'^T M_{\boldsymbol{\lambda}}')^{-1} M_{\boldsymbol{\lambda}}'^T P E(\mathbf{q}), \quad (11)$$

where we have defined (a) the *projection* matrix $P = I - A(A^T A)^{-1} A^T$ (it projects vectors to the *complement* of the subspace spanned by the columns of $A = [A_0\, 1_{N\times 1}\, A_1 \dots A_m]$) and (b) the project-out $N \times (4+n)$ motion template matrix $M_{\boldsymbol{\lambda}}' = P M_{\boldsymbol{\lambda}}$, whose columns contain the motion template images after being projected by P. Note that if we similarly define the fixed matrices $M_i' = P M_i$, then from Eq. (8) we obtain

$$M_{\boldsymbol{\lambda}}' = M_0' + \sum_{i=1}^{m} \lambda_i M_i'. \quad (12)$$

After computing $d\tilde{\mathbf{p}}$ using (11), the texture parameter update can be computed by

$$d\tilde{\boldsymbol{\lambda}}' = (A^T A)^{-1} A^T (E(\mathbf{q}) - M_{\boldsymbol{\lambda}} d\tilde{\mathbf{p}}). \quad (13)$$

Note that $(A^T A)^{-1} A^T$ is also a non-changing $(2+m) \times N$ matrix and thus can be precomputed.

After $d\tilde{\boldsymbol{\lambda}}'$ has been computed, we rectify the original texture parameterization by inverting (7), setting $d\boldsymbol{\lambda} = d\boldsymbol{\lambda}' + u_1 \boldsymbol{\lambda}$, which yields the final inverse-compositional estimates for $d\tilde{\boldsymbol{\lambda}} = [d\mathbf{u}^T, d\boldsymbol{\lambda}^T]^T$ and $d\mathbf{q} = [d\tilde{\mathbf{p}}^T, d\tilde{\boldsymbol{\lambda}}^T]^T$; note that recovering the original texture parameterization does not affect the covariance matrix $\Sigma_{\mathbf{q}}$ of Eq. (10b). We subsequently update the model parameters using the newly computed increments, as follows: (a) The m eigen-texture

parameters $\boldsymbol{\lambda}$ are updated additively, $\boldsymbol{\lambda} \leftarrow \boldsymbol{\lambda} + d\boldsymbol{\lambda}$. (b) The 2 gain/offset correction parameters u are updated inverse-compositionally, $T_{\mathbf{u}} \leftarrow T_{\mathbf{u}} \circ T_{d\mathbf{u}}^{-1}$ (simple closed form solution). (c) The $4+n$ warp parameters $\tilde{\mathbf{p}}$ are updated inverse-compositionally according to Eq. (4) – a practical algorithm to achieve this is given in [19], while in Sec. 4.1 we provide an alternative approach through computation of the inverse-compositional to forwards-additive Jacobian matrix $J_{\tilde{\mathbf{p}}}$, in which case we have $\tilde{\mathbf{p}} \leftarrow \tilde{\mathbf{p}} + J_{\tilde{\mathbf{p}}} d\tilde{\mathbf{p}}$.

In the approach we have described so far, the parameter increments $d\mathbf{q}'$ are computed from the current synthesis error image $E(\mathbf{q})$ using the mapping $K(\mathbf{q}) = \sigma^{-2}\Sigma_{\mathbf{q}} H_{\boldsymbol{\lambda}}^T$ (see Eq. (10a)) which **adapts** as the search progresses, depending to the current estimate of the texture parameter $\boldsymbol{\lambda}$. Actually the parameter updates given in Eq. (10a) are identical to those of the accurate simultaneous algorithm of [15]. However, our derivation lends itself to a more efficient implementation, since using our partitioned formulas of (11) and (13) is preferable to directly solving (10a) for the combined shape and texture vector $d\mathbf{q}'$ as is done in [15].

Moreover, the very efficient project-out algorithm of [19] can actually be derived if we use the *unadapted* motion templates M_0 which correspond to the mean texture image A_0 instead of the proper $M_{\boldsymbol{\lambda}}$, *i.e.* setting $M'_{\boldsymbol{\lambda}} = M'_0$ in Eq. (11). Our derivation thus makes clear why the project-out algorithm fails in the case of images with substantial texture variability such as the generic-face AAM studied in [15], where the previous approximation can be very crude. As we expect, the adapted motion templates for textures very different from the mean texture can significantly diverge from their unadapted versions.

Are there any algorithms that lie somewhere in-between the simultaneous and project-out algorithms, both in terms of computational cost and performance? The answer is positive: we can in general only *partially adapt* the motion templates by retaining r, $0 \leq r \leq m$ out of the m constituents of the fully-adapted motion templates of Eqs. (8) and (12), *i.e.* make the approximations

$$M_{\boldsymbol{\lambda}} \approx M_0 + \sum_{i=1}^{r} \lambda_i M_i, \quad M'_{\boldsymbol{\lambda}} \approx M'_0 + \sum_{i=1}^{r} \lambda_i M'_i \quad (14)$$

We call r the *order* of our algorithm. One can actually quantify the error incurred by this approximation to the least-squares solution using matrix perturbation analysis techniques from the numerical linear algebra literature [17], showing, for example, that the approximation causes non-zero steady-state matching error, but this is beyond the scope of the present paper. Note that the fully-adapted simultaneous algorithm corresponds to $r = m$, while the unadapted project-out to $r = 0$. Also note that, since only the $\boldsymbol{\lambda}_{1:r}$ part of $\boldsymbol{\lambda}$ is needed in Eq. (14) for computing the approximate motion templates, we need compute only the first $r+2$ elements of $d\tilde{\boldsymbol{\lambda}}'$ in Eq. (13). Hence, in computing $E(\mathbf{q})$ by Eq. (3) the sum needs only extend from 1 to r.

The computational cost of the variable-order inverse compositional algorithm will be analyzed now. For solving Eqs. (11) and (13), we must first compute $E(\mathbf{q})$, $M_{\boldsymbol{\lambda}}$, and $M_{\boldsymbol{\lambda}}'^T M'_{\boldsymbol{\lambda}}$; all other terms remain fixed and can be precomputed. Regarding $E(\mathbf{q})$, the needed first r terms in the sum of Eq. (3) can be computed, with cost $\mathcal{O}(rN)$. The approximate motion templates $M_{\boldsymbol{\lambda}}$ can be computed using Eq. (14) with cost $\mathcal{O}(rN)$. The cost of computing $M_{\boldsymbol{\lambda}}'^T M'_{\boldsymbol{\lambda}}$ straightforwardly, as is done in [15], is $\mathcal{O}(Nn^2)$. Significant savings can be obtained if we precompute the *correlation* matrices $R_{ij} = R_{ji} = M_i'^T M'_j + M_j'^T M'_i$, for $i \neq j$, and $R_{ii} = M_i'^T M'_i$, where $i, j = 0, \ldots, m$, similarly to [3]; note that each of the R_{ij} is a $(n+4)\times(n+4)$ matrix. Then:

$$M_{\boldsymbol{\lambda}}'^T M'_{\boldsymbol{\lambda}} \approx R_{00} + \sum_{i=1}^{r} \lambda_i \Big(R_{i0} + \sum_{j=1}^{i} \lambda_j R_{ij} \Big) \quad (15)$$

This means that $M_{\boldsymbol{\lambda}}'^T M'_{\boldsymbol{\lambda}}$ can be computed in $\mathcal{O}(r^2 n^2)$ cost, which usually is much better than the $\mathcal{O}(Nn^2)$ of the straightforward approach. For the adapted algorithms ($r > 0$), the system matrix $M_{\boldsymbol{\lambda}}'^T M'_{\boldsymbol{\lambda}}$ changes and thus we should add the $\mathcal{O}(n^3)$ cost of matrix inversion. Taking all contributions into account, the cost per iteration of the adapted rank-r procedure is $\mathcal{O}\big(r^2 n^2 + (n+r)N + n^3\big)$; note that for the unadapted project-out algorithm ($r = 0$) the matrix inverse can be precomputed and thus the cost per iteration is $\mathcal{O}\big(n^2 + nN\big)$.

3.2. Adaptation Through Mean Template Update

Adapting the AAM fitting matrix through the motion template adjustment technique just described compensates for the departure of the target image texture from the model's mean texture. With the order-r algorithm we are allowed utilizing the first r eigen-textures to decrease the mismatch. An alternative approach we pursue here is bringing the mean template texture A_0 itself nearer to the target images, as illustrated in Fig. 3.

By observing Eqs. (14) and (15) one realizes that the error of the order-r algorithm will be negligible if all parameters λ_i for $i > r$ can get sufficiently small. In particular, if all neglected coefficients are zero, then the algorithm is exact. This implies that for our order-r algorithm replacing the original mean texture vector A_0 with its adapted version

$$A'_0 = A_0 + \sum_{i=r+1}^{m} \lambda_i A_i \quad (16)$$

completely eliminates the approximation error. In case that we pre-compute the correlation matrices R_{ij}, we need to update those of them that depend on the mean texture A'_0

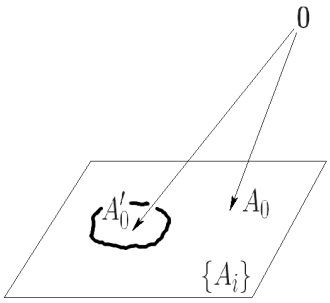

Figure 3. By updating the mean template texture from A_0 to A_0', where A_0' is nearer to the target images but still lying in the original affine space $A_0 + \text{span}\{A_i\}$, the model's representational power remains the same. The benefit is that the motion template mismatch of the order-r algorithm can be reduced.

Procedure	Complexity
Fitting iteration	$\mathcal{O}\big(r^2n^2 + (n+r)N + n^3\big)$
Template update	$\mathcal{O}\big(m(m-r)n^2\big)$

Table 1. Computational cost for a single fitting iteration and the mean template update for varying adaptation order r. N: number of texture samples; n/m: number of eigenshapes/eigentextures.

and are involved in the order-r approximation (15):

$$\begin{aligned} R'_{i0} &= R_{i0} + \sum_{j=r+1}^{m} \lambda_j R_{ij}, \quad i = 1, \ldots, r \\ R'_{00} &= R_{00} + \sum_{i=r+1}^{m} \lambda_i \Big(R_{i0} + \sum_{j=r+1}^{i} \lambda_j R_{ij} \Big) \end{aligned} \tag{17}$$

The cost of the template update procedure is thus $\mathcal{O}\big(m(m-r)n^2\big)$. Note that this cost decreases as r gets bigger and gets zero for the simultaneous algorithm ($r = m$), for which the template update has no effect. This is exactly the opposite trend to the fitting cost we studied previously, which increases with the order of the algorithm. We summarize our complexity results in Table 1.

Note that our approach should not be confused with general parametric model adaptation algorithms such as the one presented in [20]; there the goal is to change the mean texture vector A_0 and eigentextures A_i so that the texture space spanned by the model changes, adapting to the objects seen after the training phase of the model. In contrast, the mean texture update procedure we propose is just a technique that makes model fitting accurate when using low-rank algorithms, but otherwise leaves the representational power of the model unchanged.

4. Incorporating Prior Information

Active Appearance Models exhibiting considerable variability are often used in applications. For example, tracking previously unseen faces requires building generic face AAMs trained on big multi-person face datasets. These models are necessarily very diverse, comprising a large number of shape and particularly texture modes. In fitting such non-specific AAMs to images, minimizing the norm of the error image is typically inadequate, and incorporating additional prior information is crucial for regularizing the solution and improving the robustness of the method. This prior information is typically either provided by the system dynamic equation in the context of object tracking applications, or induced as a static PCA prior parameter model learnt from the training set.

Introducing prior constraints, the penalized error functional which needs to be minimized becomes

$$f(\mathbf{q}) = \frac{1}{2\sigma^2}\|E(\mathbf{q})\|_2^2 + Q(\mathbf{q}), \tag{18}$$

where the error image $E(\mathbf{q})$ is given by Eq. (3) and $Q(\mathbf{q}) = \frac{1}{2}(\mathbf{q}-\mathbf{q}_0)^T\Sigma_{\mathbf{q},0}^{-1}(\mathbf{q}-\mathbf{q}_0)$ is a quadratic penalty corresponding to Gaussian prior with mean $\mathbf{q}_0$ and covariance matrix $\Sigma_{\mathbf{q},0}$, respectively. In incremental image matching, we iteratively improve $f(\mathbf{q})$. This is straightforward for the forwards additive parameter update strategy, where one minimizes $f(\mathbf{q} + d\mathbf{q})$ over the *forwards-additive* parameter update vector $d\mathbf{q}$ [9]. In the inverse compositional case, we minimize $f(\mathbf{q} + J_{\mathbf{q}}d\mathbf{q})$ over the *inverse-compositional* parameter update vector $d\mathbf{q}$, where the Jacobian matrix $J_{\mathbf{q}}$ converts the inverse compositional parameter update to its forwards additive first-order equivalent. The parameter update formula for the maximum aposteriori $\mathbf{q}_{MAP}$ is:

$$\mathbf{q}_{MAP} \leftarrow \Sigma_{\mathbf{q},MAP}\big[\Sigma_{\mathbf{q},0}^{-1}\mathbf{q}_0 + (J_{\mathbf{q}}\Sigma_{\mathbf{q}}J_{\mathbf{q}}^T)^{-1}(\mathbf{q}_{MAP} + J_{\mathbf{q}}d\mathbf{q})\big] \tag{19a}$$

$$\Sigma_{\mathbf{q},MAP}^{-1} = \Sigma_{\mathbf{q},0}^{-1} + (J_{\mathbf{q}}\Sigma_{\mathbf{q}}J_{\mathbf{q}}^T)^{-1}, \tag{19b}$$

where the least-squares parameter update estimate $d\mathbf{q}$ and the corresponding covariance matrix $\Sigma_{\mathbf{q}}$ are the inverse-compositional domain quantities computed in Section 3.2.

4.1. Including Priors into Flexible Warp-based Inverse Compositional Algorithms

To utilize the constrained parameter update Eq. (19b), we need to compute the parameter Jacobian $J_{\mathbf{q}}$. Concentrating on the shape parameters $\tilde{\mathbf{p}}$ (since the texture parameters $\boldsymbol{\lambda}$ are updated additively), we need to compute the $(4+n)\times(4+n)$ Jacobian matrix $J_{\tilde{\mathbf{p}}}$ which maps the inverse-compositional increment $d\tilde{\mathbf{p}}$ to its additive first-order equivalent $J_{\tilde{\mathbf{p}}}d\tilde{\mathbf{p}}$. While [1] has addressed that for the simple case of globally affine warps, computing it for the more flexible warps used in AAM has not been reported before.

We start from the approximate relationship $\mathbf{W}(\mathbf{x};\tilde{\mathbf{p}} + J_{\tilde{\mathbf{p}}}d\tilde{\mathbf{p}}) \approx \mathbf{W}\big(\mathbf{W}(\mathbf{x};-d\tilde{\mathbf{p}});\tilde{\mathbf{p}}\big)$, which holds for all points $\mathbf{x}$ in the image plane to first order in $d\tilde{\mathbf{p}}$ [1, 19]. Differentiation w.r.t. $d\tilde{\mathbf{p}}$ yields

$$\frac{\partial \mathbf{W}}{\partial \tilde{\mathbf{p}}}\Big|_{(\mathbf{x};\tilde{\mathbf{p}})} J_{\tilde{\mathbf{p}}} = -\frac{\partial \mathbf{W}}{\partial \mathbf{x}}\Big|_{(\mathbf{x};\tilde{\mathbf{p}})} \frac{\partial \mathbf{W}}{\partial \tilde{\mathbf{p}}}\Big|_{(\mathbf{x};\tilde{\mathbf{p}}=\mathbf{0})}. \tag{20}$$

If we apply Eq. (20) for each landmark $\mathbf{x}_l$, $l = 1, \ldots, L$, and then solve for $J_{\tilde{\mathbf{p}}}$, we obtain the least-squares estimate

$$J_{\tilde{\mathbf{p}}} = -\left(\frac{\partial \mathbf{W}}{\partial \tilde{\mathbf{p}}}\Big|^T_{(\mathbf{x}_{1:L};\tilde{\mathbf{p}})} \frac{\partial \mathbf{W}}{\partial \tilde{\mathbf{p}}}\Big|_{(\mathbf{x}_{1:L};\tilde{\mathbf{p}})}\right)^{-1} \left(\frac{\partial \mathbf{W}}{\partial \mathbf{x}}\Big|_{(\mathbf{x}_{1:L};\tilde{\mathbf{p}})} \odot \frac{\partial \mathbf{W}}{\partial \tilde{\mathbf{p}}}\Big|_{(\mathbf{x}_{1:L};\mathbf{0})}\right), \quad (21)$$

where $\frac{\partial \mathbf{W}}{\partial \tilde{\mathbf{p}}}\Big|_{(\mathbf{x}_{1:L};\tilde{\mathbf{p}})}$ is the $(2L) \times (4+n)$ matrix stacking the derivatives evaluated on the L landmark positions in the base shape for shape parameters $\tilde{\mathbf{p}}$, and $\odot$ denotes an appropriate stacked block-by-block matrix product. Computation of $\frac{\partial \mathbf{W}}{\partial \tilde{\mathbf{p}}}$ on the landmark points is straightforward. For the $\frac{\partial \mathbf{W}}{\partial \mathbf{x}}$ terms, it can be shown that for the often used thin-plate spline warps [7] most of the cost can be moved to a pre-computation stage. Overall, computation of the Jacobian $J_{\tilde{\mathbf{p}}}$ is quite efficient, dominated by the formation and inversion of the system matrix in Eq. (21) with complexity $\mathcal{O}\left(n^2 L + n^3\right)$. Further details are given in the paper's online Appendix available from the authors' web page.

5. Experiments

To evaluate the proposed approach, we have carried out face matching experiments on both static images and video sequences using AAMs trained on multi-person datasets.

Our first set of face matching experiments on static images utilizes the frontal image sets of the XM2VTS database [21]. This dataset contains frontal images of 295 subjects; each of the subjects was photographed during 4 different sessions at approximately 1 month intervals, with 2 images acquired per session, for a total of 2360 shots. Our second set of experiments on face tracking in video sequences utilizes a 5000 frame long video of a single talking person, publicly available from the FGnet project (`http://www-prima.imag.fr/FGnet`). All images are in color and at 720x576 pixels resolution. Markup of 68 facial landmarks on all images of both the XM2VTS (manual) and talking face datasets (semi-automatic), also publicly available from FGnet, facilitates evaluating the matching performance of the different fitting algorithms.

For AAM fitting on static images, we have trained a model on 150 faces (those without facial hair/glasses) in XM2VTS's first session. We use a color AAM sampled at 3000 points, resulting in $N = 9000$ texture samples in the finest scale. Performing PCA analyses and retaining 80% of the shape variance and 95% of the texture variance has yielded $n = 11$ and $m = 87$ shape and texture modes, respectively. During fitting, 3 scales of a gaussian pyramid are used, stopping at each scale after a maximum of 10 parameter updates or when the maximum shape displacement between consequtive iterations gets less than 1 pixel. We test the performance of the different algorithms on 166 images of the second session (those without facial hair/glasses). Following [10], we systematically displace the model's mean shape from the ground-truth position by $[-20, -10, 0, 10, 20]$ pixels in the $x-$direction and similarly in the $y-$direction (25 displacements for each of the 166 images, totaling $25 \times 166 = 4150$ runs for each of the algorithms). We report the mean point-to-point error (in pixels) of each landmark of the converged shape from its ground-truth position (averaged over the L landmarks and all runs), the frequency of convergence, defined as the proportion of searches with error less than 10 pixels, the mean number of fine-scale iterations before convergence, and the speed of the search (in frames/sec). Timings refer to our Matlab implementation on a 2.2 GHz laptop computer. Some critical sub-routines have been implemented as MEX files; most notably, since AAM fitting necessitates repeated image resampling, *i.e.* computing $I(\mathbf{W}(\mathbf{x}, \mathbf{p}))$ for each iteration of the algorithm, this has been implemented using OpenGL and is GPU-accelerated. We conducted the experiment once without using a prior constraint, and once with a PCA-based prior on the shape and texture parameters. Three values for the adaptation order-r have been used, namely $r = 0$ (project-out), $r = m/2$, and $r = m$ (simultaneous). The results are summarized in Table 2. We see that on this challenging dataset using a prior constraint vastly improves the AAM fitting robustness, raising the frequency of search convergence from around 70% to nearly 99%. Increasing the order-r of motion template adaptation also improves performance, but to a smaller extent. It is also notable that constrained models also converge in fewer iterations. Regarding the efficiency of the algorithms, the computational overhead of using a prior model is milder than that of increasing the adaptation order-r.

Use Prior	Adaptation Order r	Pt-Pt Error (pix.)	Conv. Freq. (%)	Mean # Iter	Fit Speed (fps)
no	0	10.30	66.8	5.20	4.5
	$m/2$	9.12	71.6	4.79	2.6
	m	8.58	74.1	4.79	1.7
yes	0	4.84	97.2	4.05	3.0
	$m/2$	4.61	98.9	3.08	2.5
	m	4.56	98.9	3.12	1.8

Table 2. XM2VTS face matching results.

The same XM2VTS-trained AAM model described above was also utilized for face tracking on the talking face sequence. At the first frame and also whenever the search diverged (mean point-to-point error more than 10 pixels) the model was (re-)initialized with the ground truth shape, otherwise the result on the previous frame was used as initial condition. Only the finest resolution AAM model was utilized. For this experiment, we evaluate the performance of the $r = 0, m/2$ models both with and without the template update adaptation technique. When template updating

Use Prior	Adaptation		Pt-Pt Error (pix.)	Conv. Freq. (%)	Mean # Iter	Fit Speed (fps)
	Templ. Update	Order r				
no	no	0	9.49	71.3	5.67	8.1
		$m/2$	8.72	81.4	4.30	5.3
	yes	0	7.80	97.2	2.67	8.7
		$m/2$	7.99	93.2	3.84	5.6
	n/a	m	7.76	97.7	2.59	5.1
yes	no	0	6.62	98.7	2.89	6.3
		$m/2$	7.36	98.3	2.37	5.5
	yes	0	6.92	99.2	2.09	6.7
		$m/2$	6.93	99.2	2.24	5.5
	n/a	m	6.94	99.2	2.43	4.3

Table 3. Talking person face tracking results.

is enabled, we fit with the fully adapted $r = m$ model every 20 frames, and then update the template using this reliable fit result (we found that updating the template after fitting with the reduced rank model gives significantly worse results). We report in Table 3 the same summary statistics as in the previous experiment. The efficiency of the template update strategy is impressive: the $r = 0$ model in conjunction with the update strategy is almost indistinguishable in performance from the fully adapted model $r = m$, while being significantly faster than it. Moreover, since the updated $r = 0$ model also converges much more rapidly than its non-updated counterpart, it is also faster than it on average, more than amortizing the overhead of the template update step. This suggests that the combination of the very efficient $r = 0$ model with the template update strategy is particularly well-suited for real-time object tracking.

6. Conclusion

We have proposed two enhancements to inverse-compositional AAM fitting algorithms, which significantly improve the fitting performance of models exhibiting significant appearance variation, such as AAMs trained on multi-subject human face images. Especially for tracking in video, the proposed algorithms give clearly better results than previous approaches, while still being very efficient.

References

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

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

[3] A. Batur and M. Hayes. Adaptive active appearance models. *IEEE Tr. on Image Proc.*, 14(11):1707–1721, 2005.

[4] J. R. Bergen, P. Anandan, K. J. Hanna, and R. Hingorani. Hierarchical model-based motion estimation. In *Proc. Europ. Conf. on Comp. Vision*, pages 237–252, 1992.

[5] M. J. Black and A. D. Jepson. Eigentracking: Robust matching and tracking of articulated objects using a view-based representation. *Int. J. of Comp. Vision*, 26(1):63–84, 1998.

[6] V. T. Blanz and T. Vetter. Face recognition based on fitting a 3D morphable model. *IEEE Tr. on PAMI*, 25(9):1063–1074, 2003.

[7] F. L. Bookstein. Principal warps: Thin-plate splines and the decomposition of deformations. *IEEE Tr. on PAMI*, 11(6):567–585, 1989.

[8] T. F. Cootes, G. J. Edwards, and C. J. Taylor. Active appearance models. *IEEE Tr. on PAMI*, 23(6):681–685, 2001.

[9] T. F. Cootes and C. J. Taylor. Constrained active appearance models. In *Proc. ICCV*, volume 1, pages 748–754, 2001.

[10] T. F. Cootes and C. J. Taylor. An algorithm for tuning an active appearance model to new data. In *Proc. BMVC*, 2006.

[11] D. DeCarlo and D. Metaxas. Optical flow constraints on deformable models with applications to face tracking. *Int. J. of Comp. Vision*, 38(2):99–127, 2000.

[12] F. Dellaert and R. Collins. Fast image-based tracking by selective pixel integration. In *Proc. of ICCV Workshop on Frame-Rate Vision*, 1999.

[13] R. Donner, M. Reiter, G. Langs, P. Peloschek, and H. Bischof. Fast active appearance model search using canonical correlation analysis. *IEEE Tr. on PAMI*, 28(10):1690–1694, 2006.

[14] M. Gleicher. Projective registration with difference decomposition. In *Proc. CVPR*, pages 331–337, 1997.

[15] R. Gross, I. Matthews, and S. Baker. Generic vs. person specific active appearance models. *Image and Vision Comp.*, 23:1080–1093, 2005.

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

[17] N. J. Higham. *Accuracy and Stability of Numerical Algorithms*. SIAM, 2 edition, 2002.

[18] M. J. Jones and T. Poggio. Multidimensional morphable models: A framework for representing and matching object classes. *Int. J. of Comp. Vision*, 22(2):107–131, 1998.

[19] I. Matthews and S. Baker. Active appearance models revisited. *Int. J. of Comp. Vision*, 60(2):135–164, 2004.

[20] I. Matthews, T. Ishikawa, and S. Baker. The template update problem. *IEEE Tr. on PAMI*, 26(6):810–815, 2004.

[21] K. Messer, J. Matas, J. Kittler, J. Luettin, and G. Maitre. Xm2vtsdb: The extended m2vts database. In *AVBPA*, 1999.

[22] S. Romdhani and T. Vetter. Efficient, robust and accurate fitting of a 3D morphable model. In *Proc. ICCV*, pages 59–66, 2003.

[23] J. Saragih and R. Goecke. A nonlinear discriminative approach to AAM fitting. In *Proc. ICCV*, 2007.

[24] H.-Y. Schum and R. Szeliski. Construction of panoramic image mosaics with global and local alignment. *Int. J. of Comp. Vision*, 16(1):63–84, 2000.

[25] S. Sclaroff and J. Isidoro. Active blobs: region-based, deformable appearance models. *Comput. Vis. Image Underst.*, 89:197–225, 2003.

[26] R. Szeliski. Image alignment and stitching: A tutorial. *Found. and Trends in Comp. Graphics and Vision*, 2(1):1–104, 2006.