Learning AAM Fitting through Simulation[1]

Jason Saragih[*,a,b], Roland Göcke[a,c]

[a]*Research School of Information Sciences and Engineering, College of Engineering and Computer Science, The Australian National University, Canberra, ACT 0200, Australia.*
[b]*Robotics Institute, Carnegie Mellon University, Pittsburgh, USA*
[c]*Human-Computer Communication Laboratory, Faculty of Information Sciences & Engineering, University of Canberra, Canberra, ACT 2601, Australia*

Abstract

The Active Appearance Model (AAM) is a powerful method for modeling and segmenting deformable visual objects. The utility of the AAM stems from two fronts: its compact representation as a linear object class and its rapid fitting procedure, which utilizes fixed linear updates. Although the original fitting procedure works well for objects with restricted variability when initialization is close to the optimum, its efficacy deteriorates in more general settings, with regards to both accuracy and capture range. In this paper, we propose a novel fitting procedure where training is coupled with, and directly addresses, AAM fitting in its deployment. This is achieved by simulating the conditions of real fitting problems and learning the best set of fixed linear mappings, such that performance over these simulations is optimized. The power of the approach does not stem from an update model with larger capacity, but from addressing the whole fitting procedure simultaneously. To motivate the approach, it is compared with a number of existing AAM fitting procedures on two publicly available face databases. It is shown that this method exhibits convergence rates, capture range and convergence accuracy that are significantly better than other linear methods and comparable to a nonlinear method, whilst affording superior computational efficiency.

Key words: Active Appearance Model, Fitting, Discriminative, Linear Model

[1]This is the authors' preprint copy. The printed version is available at DOI: 10.1016/j.patcog.2009.04.014. This paper should be cited as:
J. Saragih and R. Goecke. Learning AAM fitting through simulation. Pattern Recognition, 42(11): 2628-2636, Nov. 2009. Elsevier.

1. Introduction

First proposed by Edwards et al. in [1], the Active Appearance Model (AAM) has attracted much interest in the computer vision community for modeling and segmenting deformable visual objects. The method makes ingenious use of linear subspaces, allowing a compact representation of both shape and texture. Coupled with an efficient fitting procedure, the AAM has found utility in many computer vision problems, ranging from face animation [2] and medical image analysis [3], to industrial vision problems [4].

The AAM's rapid fitting stems from the utility of *fixed* linear update models. Despite its simplicity, the linear update model has been shown to approximately capture the relationship between the AAM's texture residual and the optimal parameter updates [1, 5]. However, this relationship holds only loosely, as it depends on the current shape and texture parameters [6]. As such, extensions to AAM fitting have diverged into two camps. In the first, improved fitting performance is attained by building feature vectors that better adhere to a linear relationship with the parameter updates [7, 8]. Proponents of the second camp reformulate the analytic fitting problem such that the linear update model is better justified [6, 9, 10]. To reduce computational complexity, these methods usually make some assumptions about the forms of the analytic updates.

In this work, a new fitting procedure is proposed that couples training and testing through simulations of real fitting conditions. The method learns a set of *fixed* linear update models, each of which accounts for different distributions of the AAM parameters about their optimum, throughout the iterative fitting procedure. As these updates are fixed, and by virtue of their sequential application to the image, this approach affords an efficient evaluation.

The outline of this paper is as follows. In Section 2, a brief overview of related work is presented. Next, we present our approach for AAM fitting in Section 3, addressing implementation complexities in Sections 3.1 and 3.2.

[*]Corresponding author

Email addresses: `jsaragih@andrew.cmu.edu` (Jason Saragih), `roland.goecke@ieee.org` (Roland Göcke)

Empirical evaluations are presented in Section 4. We conclude in Section 5 with additional remarks and directions for future work.

2. Background

The AAM models intrinsic variations in shape and texture of deformable visual objects as a linear combination of basis modes of variation that are composed with a global transformation[2]:

$$S(\mathbf{q}_s)\colon \Re^{n_s} \mapsto \Re^{2n} = s(\mathbf{I} \otimes \mathbf{R})(\boldsymbol{\mu}_s + \boldsymbol{\Phi}_s \mathbf{p}_s) + \mathbf{1} \otimes \mathbf{t} \tag{1}$$

$$T(\mathbf{q}_t)\colon \Re^{n_t} \mapsto \Re^{m} = \alpha(\boldsymbol{\mu}_t + \boldsymbol{\Phi}_t \mathbf{p}_t) + \beta \mathbf{1}, \tag{2}$$

where S and T denote the generative models for shape and texture, parameterized by $\mathbf{q}_s = \{s, \mathbf{R}, \mathbf{t}, \mathbf{p}_s\}$ and $\mathbf{q}_t = \{\alpha, \beta, \mathbf{p}_t\}$ respectively. Here, $\{\boldsymbol{\mu}_s, \boldsymbol{\Phi}_s\}$ and $\{\boldsymbol{\mu}_t, \boldsymbol{\Phi}_t\}$ denote the mean and basis of variations of the shape and texture, which are typically obtained by applying PCA on the training data. The intrinsic shape is composed with a similarity transform, parameterized by a global scaling s, a rotation $\mathbf{R}$ and a translation $\mathbf{t}$. The intrinsic texture is scaled by a global gain α and biased by β. Finally, n denotes the number of points in the model's shape and m denotes the number of pixels in the model's texture.

AAM fitting is the process of finding the model parameters $\mathbf{p} = \{\mathbf{q}_s, \mathbf{q}_t\}$ which best fit an AAM to an image I. This is usually an iterative procedure that sequentially updates the model parameters $\mathbf{p}$ through an update function:

$$\Delta\mathbf{p} = U(\Diamond; \mathbf{p}) \circ F(I; \mathbf{p}). \tag{3}$$

Here, F is a feature extraction function that represents the image I from the perspective of the AAM at its current parameter settings and $\Delta\mathbf{p}$ are the updates to be applied to the current parameters. U is typically chosen to be the linear update model:

$$U(\mathbf{f}; \mathbf{p})\colon \Re^{m} \mapsto \Re^{n_s + n_t} = \mathbf{G}\mathbf{f} + \mathbf{b} \quad \text{where} \quad \mathbf{f} = F(I; \mathbf{p}), \tag{4}$$

[2]**Notation:** Vectors are written in lowercase bold and matrices in uppercase bold, where $\mathbf{1}$ denotes the all one vector and $\mathbf{I}$ the identity matrix. Greek letters denote either vectors or matrices depending on context. The Kronecker (tiling) product is written using $\otimes$. The vec$\{.\}$ operator vectorizes a matrix by stacking its columns. Functions are written in upper case with $\circ$ denoting their composition. When composing functions with multiple parameters, $\Diamond$ denotes a place-holder, i.e.: $A(B(\mathbf{x}); \mathbf{y}) = A(\Diamond; \mathbf{y}) \circ B(\mathbf{x})$.

although nonlinear mappings have also been used [11, 12]. In any case, a good coupling between U and F is required to ensure good predictions of the updates. There are two general approaches to AAM fitting: *generative* and *discriminative*. In the following, we will briefly discuss each in turn.

2.1. Generative Fitting

$$Q(\mathbf{p}) = \|T(\mathbf{q}_t) - I \circ W \circ S(\mathbf{q}_s)\|^2, \tag{5}$$

where W is a function that warps pixel locations in the reference frame onto the image. Robust variants of this objective have been used also, in which case the problem takes the form of a continually reweighted least squares problem [13]. In general, the Gauss-Newton method has become the optimizer of choice, where at each iteration the parameters are updated using the linear form in Equation (4). However, a naive implementation is computationally expensive as $\{\mathbf{G}, \mathbf{b}\}$ depends on $\mathbf{p}$. As such, most generative approaches to AAM fitting either assume some parts of the update computation are fixed or reformulate the problem such that they are.

The original generative approach in [5] assumes that the Jacobian of Equation (5) is fixed, allowing $\{\mathbf{G}, \mathbf{b}\}$ to be precomputed. More recently, in a method coined Adaptive AAM [10], the fixed Jacobian assumption is relaxed by decomposing it and assuming only the component pertaining to the derivative of the warping function is fixed. The resulting method exhibits improved accuracy; however, as the linear update model depends on the current texture parameters, the fitting procedure is computationally expensive.

Adaptations of the inverse-compositional image alignment [14] to AAM fitting has also attracted some interest. By reversing the roles of the image and the model in the fitting objective, the derivative of the warping function is fixed, resulting in significant computational savings. The project-out method [6], for example, minimizes the cost in a subspace orthogonal to the modes of texture variation, resulting in an analytically fixed linear update model. Despite being the fastest method to date, it works well only for objects exhibiting small amounts of variation, such as, for example, in person-specific face tracking [15]. This problem is partially addressed by the simultaneous method [9], which solves for the shape and texture parameters simultaneously. However, similar to the method in [10], its update model depends on the current texture parameters, again resulting in a computationally expensive fitting procedure.

2.2. Discriminative Fitting

Although generative approaches have an intuitive appeal, they are optimally constructed for synthesis, rather than fitting. To address this drawback, a number of authors propose learning a fitting strategy discriminatively, where both aligned and misaligned examples are considered during training [1, 7, 12, 16, 17]. There are two general strategies in discriminative fitting. The first is to learn an objective function with desirable properties that promote convergence of a gradient descent type search to the desired optimum [16, 17]. The second is to learn a fixed mapping between the features $F(I;\mathbf{p})$ and the parameter updates $\Delta\mathbf{p}$ [1, 7, 12], given a training set of perturbed model parameters:

$$\{F(I_i;\mathbf{p}_i^* - \Delta\mathbf{p}_i)\ ,\ \Delta\mathbf{p}_i\}_{i=1}^{d}\,, \tag{6}$$

where $\mathbf{p}_i^*$ is the optimal parameter setting for the i^{th} sample and d is the total number of perturbations in the training set. The advantage of the second approach is that no nonlinear optimization is required during fitting, potentially leading to an efficient fitting procedure.

In the original AAM formulation [1], the linear update model was shown to approximately explain the relationship between the AAM's texture residual:

$$F(I;\mathbf{p}) = T(\mathbf{q}_t) - I \circ W(\mathbf{q}_s) \tag{7}$$

and the parameter updates $\Delta\mathbf{p}$, around the optimal parameter settings $\mathbf{p}^*$ for a given image. The update U is easily found through linear regression on the data set in Equation (6). The direct appearance model method [8] affords better memory efficiency during training as well as improves fitting performance by using the PCA reduced texture residuals and predicts the shape directly from the texture. In [7], a linear relationship is learnt between the canonical projections of the texture residuals and parameter updates. The method utilizes canonical correlation analysis to find the subspaces which best adheres to a linear relationship. In [12], the mapping capacity is increased by using a boosted mapping function of weak learners.

Finally, it should be noted that a number of discriminative approaches to template tracking have been proposed recently with similar forms to the deformable model fitting methods described above. Examples of this include [18, 19, 20]. However, the application of such approaches to deformable model fitting requires a specialized treatment due to the higher dimensionality of the motion model and the typically larger variability of intrinsic appearance.

3. Learning the Mappings through Simulation

The major drawbacks of current AAM fitting methods are

- A fixed linear update model cannot accurately account for the various error terrains about the optimum in different images.
- Adapting the update model to the image at hand requires a costly process of re-calculating it for every iteration.
- Increasing the complexity of the mapping function in a discriminative setting may lead to issues with generalizability as well as computational complexity.

In this work, the drawbacks of a fixed linear model are addressed whilst maintaining efficiency. To this end, we propose learning the entire fitting procedure in a discriminative framework rather than extracting the update model, or an approximation thereof, from a generative stand point. Learning is performed on examples of real fitting scenarios, simulated on the training data.

Consider the class of linear update models that utilize the additive parameter update: $\mathbf{p} \leftarrow \mathbf{p}+\Delta\mathbf{p}$. If no adjustments to the parameters are performed between iterations, such as projections onto the three standard deviations ellipsoid, then the effective parameter adaptation after k iterations is given by:

$$\mathbf{p} \leftarrow \mathbf{p} + \sum_{i=1}^{k} \Delta\mathbf{p}_i. \tag{8}$$

Utilizing a different update model in each iteration, the parameter updates can be written as:

$$\Delta\mathbf{p}_i = \mathbf{G}_i \; F\left(I \; ; \; \mathbf{p} + \sum_{j=1}^{i-1} \Delta\mathbf{p}_j\right) + \mathbf{b}_i, \tag{9}$$

where $\{\mathbf{G}_i, \mathbf{b}_i\}$ is the *fixed* update model for the i^{th} iteration. Given the training set in Equation (6), the optimal update models for all k iterations can be found by minimizing a cost function of the form:

$$Q(\mathbf{m}) = \frac{1}{d} \sum_{j=1}^{d} C_D\left(I_j, \mathbf{p}_j, \mathbf{s}_j^* \; ; \mathbf{m}\right), \tag{10}$$

where $\mathbf{s}_j^*$ are the hand-labeled annotations for the j^{th} training sample: $\mathbf{p}^* = \min_{\mathbf{q}_s} \|\mathbf{s}^* - S(\mathbf{q}_s)\|^2$. The parameters of this cost function are the update models themselves:

$$\mathbf{m} = \left[\text{vec}(\mathbf{G}_1); \ldots; \text{vec}(\mathbf{G}_k); \mathbf{b}_1; \ldots; \mathbf{b}_k\right]. \tag{11}$$

The distance function C_D in Equation (10) penalizes the difference between the manually annotated shapes and the predicted model's shape after N_i iterations[3]:

$$C_D\left(I, \mathbf{p}, \mathbf{s}^*; \mathbf{m}\right) = \frac{1}{2}\left\| S\left(\mathbf{p} + \sum_{i=1}^{k} \Delta\mathbf{p}_i\right) - \mathbf{s}^*\right\|^2. \tag{12}$$

Compared to texture based error measures, commonly used in generative AAM fitting, this distance function better encompasses all available knowledge about the optimal parameter setting, i.e. the hand-labeled annotations. With this formulation, the training procedure essentially simulates real fitting problems on the set of training images and perturbations. If at deployment, the unseen images and their perturbations resemble those in the training set, then the fitting performance of the minimiser of Equation (10) can be expected to approach that at training.

Having trained the update models that minimize Equation (10), AAM fitting then proceeds as outlined in Algorithm 1. Notice that no checks need to be made regarding the reduction of texture error or the magnitude of the parameter updates. Fitting is simply performed for all trained iterations with no early termination.

Compared to the training procedure of generative methods, finding the optimal update models by minimizing Equation (10) presents a number of difficulties. Firstly, the cost function is inherently nonlinear with many local minima, due to the composition of the updates with the feature vectors and the nonlinear relationship between the pixel intensities and the warping parameters. Secondly, standard numerical optimization techniques are not computationally practical for this problem. Since the parameter updates at each iteration depend on the update models for all previous iterations, the analytic gradient of Equation (12) is in general extremely complex, resulting

[3]Note that, here, we consider mappings only over the shape parameters such that $\mathbf{p} = \mathbf{q}_s$.

Algorithm 1 Discriminative Iterative AAM Fitting

```
Require: $I$, $\{U_1, \ldots, U_k\}$ and $\mathbf{p}$
1: for $i = 1$ to $k$ do
2:     $\mathbf{f} = F(I; \mathbf{p})$ {Get feature vector}
3:     $\Delta\mathbf{p} = \mathbf{G}_i\mathbf{f} + \mathbf{b}_i$ {Calculate updates}
4:     $\mathbf{p} \leftarrow \mathbf{p} + \Delta\mathbf{p}$ {Update current parameters}
5: end for
6: return $\mathbf{p}$
```

in an impractical computational burden. This matter is made worse by the potentially large number of training samples required to ensure good generalizability of the update models. In the following sections, we discuss methods to make the optimization computationally tangible as well as allowing a good *local* solution to the nonlinear problem in Equation (10) to be obtained.

3.1. Stochastic Gradient Descent

The training set in Equation (6) is generally obtained by sampling from a distribution that approximates the true distribution of perturbation errors encountered in deployment [12, 17]. Since the training set consists of an unbounded number of samples for each training image, we recast the problem as one of online estimation, where each sample, or a small set of them, is treated as it arrives. Online estimation is essentially a stochastic optimization problem, since only noisy objective function evaluations are available due to restrictions on the sample set size at each instance. There are a number of advantages of online/stochastic learning over batch learning[4]. For example, it has been widely reported that it can reduce training time [21], has the ability to escape from *shallow* local minima [22], minimizes the true risk rather than the empirical risk [23], requires far less memory, and it does not require the number of samples to be chosen a priori.

Given the instantaneous cost function at time t:

$$C_t(\mathbf{m}) = \frac{1}{n_b} \sum_{j=1}^{n_b} C_D\left(I_j, \mathbf{p}_j, \mathbf{s}_j^*;\ \mathbf{m}\right), \tag{13}$$

where n_b is the *mini-batch* size, stochastic optimization then proceeds by first

[4]Batch learning minimizes the empirical risk over the whole training set simultaneously.

calculating the gradient:

$$\mathbf{d}_t = \frac{1}{n_b} \sum_{j=1}^{n_b} \frac{\partial C_{D_j}}{\partial \mathbf{m}} \tag{14}$$

then updating the model parameters in a steepest descent fashion:

$$\mathbf{m}_{t+1} = \mathbf{m}_t - \eta_t \mathbf{d}_t, \tag{15}$$

where the step size η is scheduled such that:

$$\eta_t > 0 \quad ; \quad \eta_t \to 0 \text{ as } t \to \infty \quad ; \quad \sum_{t=0}^{\infty} \eta_t = \infty \quad \text{and} \quad \sum_{t=0}^{\infty} \eta_t^2 < \infty \tag{16}$$

to guarantee convergence to a local minimum. However, the deterministic gradient of Equation (13) cannot be trivially evaluated due to the dependence of the updates on those of previous iterations. To see this, note that the deterministic gradient of C_D is given by:

$$\frac{\partial C_D}{\partial \mathbf{m}} = \frac{\partial C_D}{\partial \mathbf{z}} \frac{\partial \mathbf{z}}{\partial \mathbf{m}} \tag{17}$$

where $\mathbf{z} = [\Delta \mathbf{p}_1; \ldots ; \Delta \mathbf{p}_{N_i}]$ is the concatenation of the parameter updates of all iterations. The derivative $\frac{\partial C_D}{\partial \mathbf{z}}$ can be easily computed from Equations (12) and (1). Now, consider the simplest case, where:

$$F(I; \mathbf{p}) = I \circ W \circ S(\mathbf{p}). \tag{18}$$

With this feature vector, the derivative of the j^{th} parameter update with respect to the bias vector of the k^{th} iteration is given by the following recursive form:

$$\frac{\partial \Delta \mathbf{p}_j}{\partial \mathbf{b}_k} = \begin{cases} \mathbf{0} & \text{if } j < k, \\ \mathbf{I} & \text{if } j = k, \\ \mathbf{G}_j \frac{\partial F}{\partial \mathbf{p}} \left(\sum_{i=1}^{j-1} \frac{\partial \Delta \mathbf{p}_i}{\partial \mathbf{b}_k} \right) & \text{if } j > k, \end{cases} \tag{19}$$

where the derivative of the warped image $\frac{\partial F}{\partial \mathbf{p}}$ is evaluated at $\mathbf{p} + \sum_{i=1}^{j-1} \Delta \mathbf{p}_i$. The derivative with respect to the k^{th} gain matrix is given by:

$$\frac{\partial \Delta \mathbf{p}_j}{\partial \mathbf{G}_k} = \frac{\partial \Delta \mathbf{p}_j}{\partial \mathbf{b}_k} \otimes F \left(\mathbf{p} + \sum_{i=1}^{k-1} \Delta \mathbf{p}_i \right). \tag{20}$$

The evaluation of these partial derivatives is computationally intensive and memory demanding. Furthermore, the complexity of evaluating these partials grows exponentially with the number of iterations k. In fact, we found that for $k > 3$, the optimization becomes impractical on typical hardware. This problem is amplified with the use of more sophisticated feature vectors, especially those that involve an adaptive normalization component.

To avoid the costly computation of deterministic gradients, we utilize simultaneous perturbation stochastic approximation (SPSA) [24]. This method computes stochastic gradients of a function with as little as two function evaluations, allowing an efficient optimization procedure to be constructed. Specifically, the stochastic gradient of Equation (13) is computed as follows:

$$\hat{\mathbf{d}}_t = \frac{1}{2\gamma_t}\left[C_t(\mathbf{q} + \gamma_t \boldsymbol{\Delta}_t) - C_t(\mathbf{q} - \gamma_t \boldsymbol{\Delta}_t)\right]\boldsymbol{\Delta}_t, \tag{21}$$

where $\boldsymbol{\Delta}_t$ is a vector of random perturbations for each model parameter, independently generated from a zero-mean probability distribution satisfying the conditions in [25], for example the Bernoulli ± 1 distribution with probability of 0.5 for each outcome. The perturbation magnitude γ_t requires similar conditions to η in Equation (16) to guarantee convergence. Common choices for η_t and γ_t, which satisfy these conditions, are:

$$\eta_t = \frac{\eta}{(a+t)^{\alpha}} \quad \text{and} \quad \gamma_t = \frac{\gamma}{t^{\beta}}, \tag{22}$$

where (η, γ) are the initial values and (α, β) are the decay rates for the step size and perturbation magnitudes, respectively. Here, a is a stabilizing constant that allows a larger initial step size η, whilst avoiding instabilities in early iterations. We should note here that since the elements of $\mathbf{m}$ have very different magnitudes, depending on which element of $\mathbf{p}$ they map onto, they are first scaled according to the maximum expected perturbation of their respective AAM parameter $\mathbf{p}$ before applying the update in Equation (15). Finally, the performance of SPSA depends on the choices made for the free variables $(\eta, \gamma, a, \alpha, \beta)$. A practical guide for choosing these variables is given in [26].

By performing online estimation using SPSA, the noise in our system is amplified: the noisy gradient approximation is made on noisy cost evaluations C_t. As such, it is important to choose the SPSA parameters such that the fluctuations due to noise do not overwhelm the optimization process and cause instability. Estimation noise can be further reduced by taking an average of a number of stochastic gradient approximations [24]. In practice, optimization is run for a number of iterations on different parameter settings, choosing the one which induces the best efficiency per update, without making the optimization unstable.

3.2. Spread Minimized Initialization

Although noise in the stochastic estimation helps the optimization escape from local minima with shallow terrain, it may still terminate in a poor local minimum. A good initialization for the update models is, therefore, required to encourage good solutions to be obtained by the optimization process. For this, consider first that we utilize only linear models at every iteration. Therefore, the performance of this method is limited by the estimation capacity of the linear model in the final iteration. However, as all update models are learnt simultaneously, all update models except the last one act as *sample redistributors* such that the distribution of the samples around their optimum in the last iteration can be well accounted for by the final linear update model.

With this in mind, we cite the results by Cootes et al. in [5], that the relationship between the texture residuals and the parameter updates is close to linear only within a small region around the optimum of each AAM parameter. In fact, this relationship has been shown to persist, though to a lesser extent, even for the simple warped texture feature [27]. As such, we argue that a reasonable initialization for the optimization in Section 3.1 is one that reduces the spread of the samples about their optimum.

To this end, we propose performing a greedy learning process for each iteration, in a sequential manner, each of which aims to minimize the spread of samples about their respective optimum, given the samples that have been updated by previous iterations. We achieve this by learning a linear regressor for every parameter independently, by minimizing the following cost function for each:

$$C_{\text{greedy}}(\mathbf{g}, b) = \frac{1}{d}\sum_{j=1}^{d} C_{\text{asy}}(\mathscr{I}_j, \mathbf{p}_j, \Delta p_j; \mathbf{g}, b) + \frac{\lambda}{2}\|\mathbf{g}\|^2, \tag{23}$$

where λ is a regularization constant and:

$$C_{\text{asy}}(\mathscr{I}, \mathbf{p}, \Delta p; \mathbf{g}, b) = \frac{1}{\epsilon - (\mathbf{g}^T\mathbf{f} + b - \Delta p)^2} \quad . \tag{24}$$

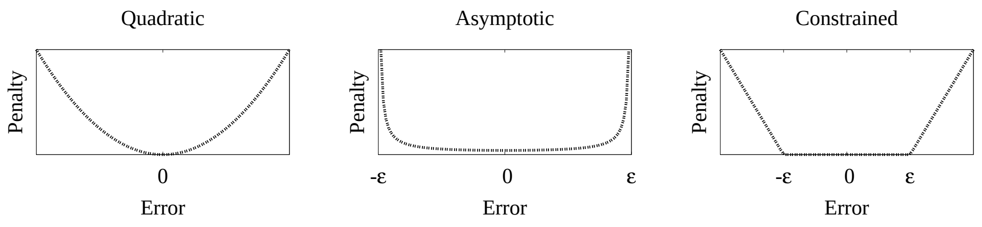

Figure 1: Quadratic, asymptotic and constrained penalizers. The asymptotic penalizer places more weight on samples far from the optimum compared to a quadratic penalizer. Compared to the constrained penalizer, the asymptotic penalizer minimizes the spread of samples rather than their error margin only.

Here, $\mathbf{f}$ is as in Equation (4) and:

$$\epsilon \;=\; \max(\Delta p_j^2) + \delta \quad ; \quad j \in \{1, \ldots, k\} \;,\; \delta \in \Re^+. \tag{25}$$

Note that the full update model is composed of the regressor for each parameter:

$$\mathbf{G} = \left[\mathbf{g}_1^T; \ldots; \mathbf{g}_{n_s}^T\right] \quad \text{and} \quad \mathbf{b} = [b_1; \ldots; b_{n_s}]\,, \tag{26}$$

where n_s is the total number of parameters.

Previously, we used a similar cost function as in Equation (23) to train nonlinear parameter update mappings in [11]. It asymptotically penalizes the distance of each sample from its optimum, placing more emphasis on samples with large perturbations compared to, for example, the quadratic loss (see Figure 1). Furthermore, this asymptotic penalizer is convex within the convex set:

$$\left\{(\mathbf{g}, b) \middle| -\sqrt{\epsilon} \;<\; \mathbf{g}^T\mathbf{f}_j + b - \Delta p_j \;<\; \sqrt{\epsilon}\,; \quad j = 1, \ldots, d\right\}, \tag{27}$$

i.e. the intersection of d convex sets, each composed of the region between two parallel hyperplanes. Due to the choice of ϵ in Equation (25), the null model ($(\mathbf{g}, b) \leftarrow \mathbf{0}$) lies within this convex region. As such, starting with the null model and performing steepest descent with a line search allows the globally optimum update model to be found. Unlike the optimization problem discussed in Section 3.1, the gradient of this cost function is easily computed: $\frac{\partial C_{\text{asy}}}{\partial b} = \theta$ and $\frac{\partial C_{\text{asy}}}{\partial \mathbf{g}} = \theta\mathbf{f}$, where:

$$\theta = 2r\left(\epsilon - r^2\right)^{-2} \quad ; \quad r = \mathbf{g}^T\mathbf{f} + b - \Delta p. \tag{28}$$

We previously proposed a similar greedy learning objective in [28] using a constrained objective (see Figure 1). However, that objective does not penalize samples within the margin, which may lead to the clustering of samples around the margin.

For optimization, we use the limited memory BFGS algorithm (L-BFGS) [29], a variant of the quasi-Newton optimizer BFGS, which avoids the cost of storing and updating the estimate of the cost function's Hessian inverse. Given the L-BFGS step direction $\mathbf{d}$, the line search is performed by solving:

$$\alpha^* = \min_{\alpha} \frac{1}{d} \sum_{j=1}^{d} \frac{1}{\epsilon - (\alpha[\mathbf{f}_j\ 1]\mathbf{d} + \mathbf{g}^T\mathbf{f} + b - \Delta p_j)^2} + \lambda \|\mathbf{d}\|^2 \alpha$$

$$\text{subject to} \quad 0 \leq \alpha \leq \min\left(\frac{\pm\sqrt{\epsilon} - \mathbf{g}^T\mathbf{f}_j - b + \Delta p_j}{[\mathbf{f}_j\ 1]\mathbf{d}} \right); j \in \{1, \ldots, d\} \qquad (29)$$

where the sign of $\sqrt{\epsilon}$ is chosen to represent the asymptote in the direction of the update. The training procedure is summarized in Algorithm 2.

4. Experiments

We conducted experiments on two publicly available face databases: the IMM Face database [30] and the XM2VTS database [31]. The simulation learnt approach (SL) was compared against the fixed Jacobian method (FJ) [5], project-out inverse compositional method (POIC) [6], the efficient estimate of the simultaneous inverse compositional method (SIC) [9] and the nonlinear discriminative method (ND) [11]. The performance of all methods was compared on these two databases, evaluated using 4-fold cross-validation, where each database was partitioned into four parts, in such a way, that test images contain unseen subjects only.

4.1. Databases

The IMM face database consists of 240 images of 40 subjects, exhibiting variations in pose, expression and lighting. Manual annotations of 58-landmarks are supplied with the database. The XM2VTS database consists of 2360 frontal images of 295 subjects with large inter-subject variability, including glasses, facial hair and race. A 68-landmark annotation of this database is publicly available from [32]. In both databases, we retained 95% of the intrinsic shape and texture variation. This gave an average of 19 modes

Algorithm 2 Greedy Spread Minimisation

```
Require: $k$ and $d$
1:  for $i = 1$ to $k$ do
2:    $\{I_j, \mathbf{s}_j^*, \mathbf{p}_j\}_{j=1}^d$ {Sample training data}
3:    for $j = 1$ to $d$ do
4:      $\mathbf{f}_j = F(\mathscr{I}_j, \mathbf{p}_j)$ {Get feature vector}
5:      for $l = 1$ to $i - 1$ do
6:        $\mathbf{p}_j \leftarrow \mathbf{p}_j + \mathbf{G}_l\mathbf{f}_j + \mathbf{b}_l$ {Update parameters}
7:      end for
8:      $\mathbf{f}_j = F(I_j, \mathbf{p}_j)$ {Get feature vector}
9:    end for
10:   Calculate $\{\epsilon_1, \ldots, \epsilon_{N_p}\}$ {Eqn. (25)}
11:   $(\mathbf{g}, b)_l \leftarrow \mathbf{0}$ for $l \in \{1, \ldots, n_s\}$ {Initialize}
12:   while !converged($\mathbf{G}_i$,$\mathbf{b}_i$) do
13:     for $l = 1$ to $n_s$ do
14:       Compute L-BFGS step $\mathbf{d}_l${See [29]}
15:       Perform line search to get $\alpha_l^*$ {Eqn. (29)}
16:       $(\mathbf{g}, b)_l \leftarrow [\mathbf{g}; b]_l + \alpha_l^*\mathbf{d}_l$ {Update model}
17:     end for
18:   end while
19:   Assign update of $i^{\text{th}}$ iteration {Eqn. (26)}
20: end for
21: return $\mathbf{G}_1, \ldots, \mathbf{G}_k$ and $\mathbf{b}_1, \ldots, \mathbf{b}_k$
```

of shape variation and 99 texture modes for the IMM database. The models built for the XM2VTS database exhibited on average 47 shape modes and 352 texture modes.

4.2. Simulation Training

To generate the training set in Equation (6), the AAM was randomly perturbed from its optimal parameter setting in each training image within $\pm 10°$ of rotation, ± 0.1 global scaling, ± 20 pixels translation and ± 1.5 standard deviations for the non-rigid shape parameters. This distribution of samples was chosen to mimic the initialization capacity of a generic detector. SL was trained for $k = 10$ iterations using the normalized warped texture feature:

$$F(I; \mathbf{p}) = N \circ I \circ W \circ S(\mathbf{p}), \tag{30}$$

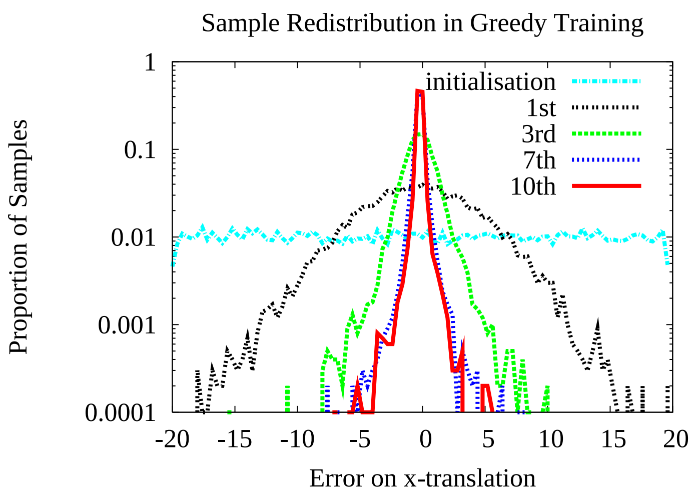

Figure 2: Redistribution of samples about the x-translation optimum in the greedy training procedure on the IMM database. Legend denotes iteration.

where N normalizes the warped texture to a mean of zero and a variance of one. The trained AAM drives only the shape parameters during fitting.

4.2.1. Greedy Training

For model initialization, we used $d = 2000$ samples in the procedure outlined in Algorithm 2. Figure 2 illustrates the redistribution of samples throughout the greedy training process for the x-translation. A similar trend can be observed for the other parameters. As described in Section 3.2, the form of Equation (23) places more emphasis on samples exhibiting large error. This is reflected in the figure where the spread of the samples about their optimum is reduced at each iteration. Notice that the reduction of sample spread between the 7th and 10th iteration is marginal, suggesting that the capacity of linear models has been exhausted, justifying the choice of $k = 10$.

4.2.2. Stochastic Optimization

The stochastic optimization of all linear update models requires an appropriate choice of mini-batch size and gradient estimates such that the noise in the stochastic approximation does not make the optimization unstable. Empirically, we found that a mini-batch size of $n_b = 50$ and $n_g = 2$ gradient estimates gave a good balance between evaluation cost and convergence rate for models in both databases. These values were the smallest for which the optimization remained stable. The step and perturbation decay rates were set to $\alpha = 0.602$ and $\beta = 0.101$, in accordance with suggestions in [26] (i.e. generic values for SPSA that are not specific to our problem). The initial perturbation size γ was chosen such that the average shape perturbation due to the update model perturbations was around one pixel, a good choice for numerical differentiation on image based functions. The initial step size was set to $\eta = 1.0^{-5}$ with the resulting model update in Equation (15) scaled according to the maximum magnitude of perturbation for the respective parameters.

4.3. Fitting Performance

To compare the five methods, the AAM parameters were randomly perturbed from their optimal settings 100 times for each test image in both databases for each of the four trials. The perturbations were sampled as described in Section 4.2, with an additional ± 0.1, ± 20 and ± 1.5 standard deviations for the global lighting gain, lighting bias and the intrinsic texture parameters, respectively. FJ, POIC and SIC were fitted using 3 levels of a Gaussian pyramid to reduce their sensitivity to local minima. SL and ND were fitted to the highest resolution image only, by virtue of their discriminative framework. For SIC, we implemented its efficient approximation proposed in [9], where the Gauss-Newton Jacobian (and, hence, the update models) are precomputed at $\mathbf{q}_t = \mathbf{0}$, since the large fitting times for the full approach were impractical for the size the models in our experiments.

The convergence performance of all five methods in both databases over all four trials is presented in Figure 3. Plots (a) and (c) illustrate the distribution of the accuracy of the converged trials whereas the average convergence accuracy of the five methods, with respect to their initial perturbations magnitudes, are given in Plots (b) and (d). The convergence rates and average fitting times for each method are shown in the legend. In all fitting trials, we declare convergence if the final point-to-point RMS error is smaller than at initialization (i.e. the fitting procedure did not diverge). The reported fitting times were obtained from running code, implemented in C++, on a 3GHz machine with 1GB of RAM, and does not include the time taken to build the Gaussian pyramids for FJ, POIC and SIC.

On both databases, the overall performance of SL is significantly better than FJ, POIC and SIC, both in convergence frequency and accuracy, and comparable to ND. Out of the three generative methods tested, FJ performed the best, achieving a smaller *best error* compared to SL on the IMM

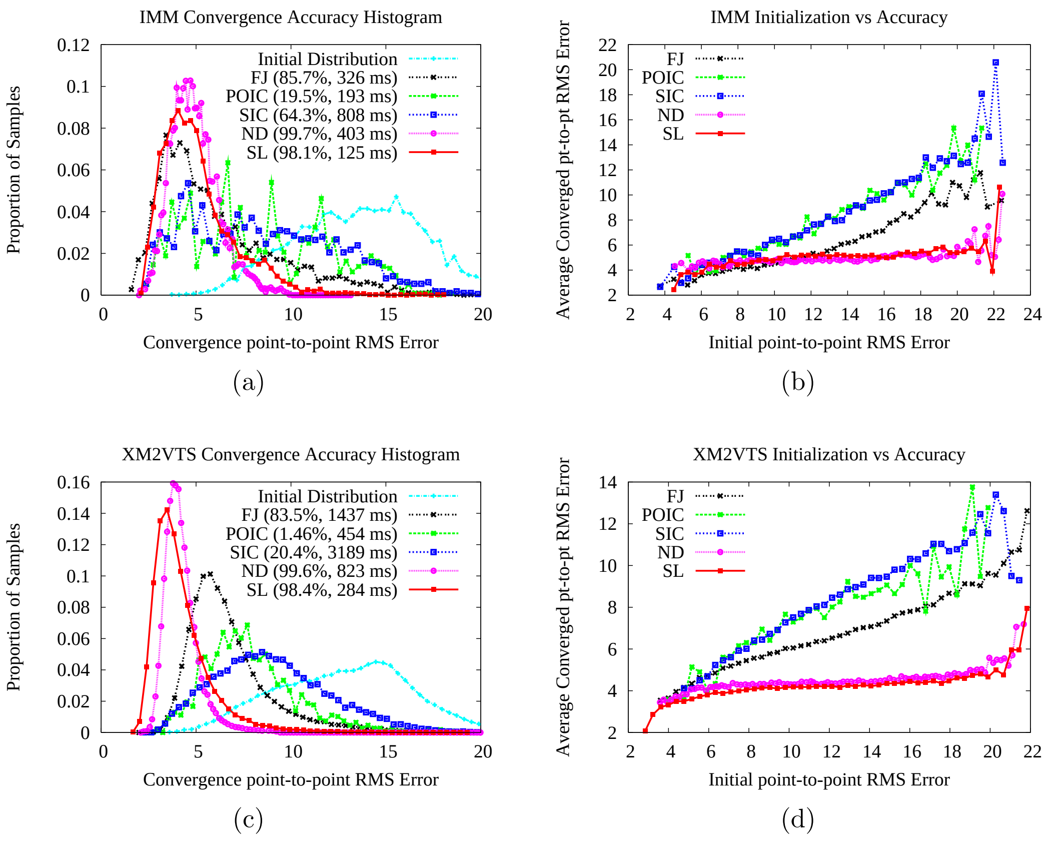

Figure 3: Convergence performance of the fixed Jacobian (FJ), project-out (POIC), simultaneous inverse compositional (SIC) and the iterative-discriminative methods (SL, ND) on the IMM and XM2VTS databases. Plots (a) and (c): histograms of accuracy at convergence (percentage in legend is the convergence frequency). Plots (b) and (d): initial error vs. average convergence accuracy.

database. However, the proportion of samples which achieve this best error is relatively small. Overall, it exhibits a larger proportion of samples at higher convergence errors above 6-pixels RMS compared to SL. On the XM2VTS database, the superiority of SL is even more pronounced. Furthermore, in both databases, the convergence frequency of SL is much higher than FJ, achieving a reduction in shape RMS error in almost every trial. Compared to POIC and SIC, again SL boasts a significant performance boost, both in convergence accuracy and frequency. This is especially evident in the XM2VTS database where the models exhibit a large number of parameters. Although the inapplicability of POIC for generic face fitting has been previously discussed in [15], results here show that the efficient approximation of SIC also performs poorly in this setting. We suspect the reason for this is to do with the large number of parameters involved in the optimization, as SIC uses an independent appearance model, resulting in a higher likelihood of getting trapped in local minima than FJ, for example. Although a true SIC implementation may improve results, the fitting times required by this method are impractical for the size of the models used for generic face fitting. Even its efficient approximation, which uses a fixed update model, is impractically inefficient. From initial experiments, we found that a full SIC implementation can further increase the processing time by a factor of 10.

Examining Plots (b) and (d) of Figure 3, the reason for the significant performance improvement achieved by SL compared to the generative methods becomes clear. The average convergence accuracy of FJ, POIC and SIC deteriorates the further initialization is from the optimum. This is due to their generative fitting regimes, which get trapped in local minima, despite their application on a Gaussian pyramid. In contrast, the deterioration of SL, which was trained discriminatively on simulations of real fitting problems, is not as dramatic, maintaining a good average convergence accuracy up to a capture range of around 20 pixels RMS, which was the maximum translation perturbation used in the training.

The performance of SL and ND in these experiments are comparable. Although the final distribution of samples achieved by ND is more compact it fails to achieve the same levels of accuracy as SL. Despite using a more sophisticated discriminative mapping function, its training regime is suboptimal. The boosting procedure over Haar-like features is essentially a greedy training procedure. As such, the trained model for each iteration significantly overfits the data, requiring the next iteration to handle the poorly predicted samples.

Despite the significant discrepancy in performance between SL and the generative methods, this performance boost is attained without sacrificing fitting efficiency. This is in contrast to other methods which either sacrifice accuracy for efficiency or vice-versa. On a per-iteration basis, SL is slightly faster than POIC, currently the fastest method to date, since it uses the same update form and drives the same number of parameters, but updates the parameters additively rather than in an inverse compositional fashion. Comparisons of overall fitting speeds depend on the initial perturbation magnitude as well as the error terrain between the initial parameter settings and the optimum. Whereas SL fitting is performed for all the iterations it was trained for, generative fitting procedures generally have an early termination criterion, judged by some measure of parameter convergence. In our experiments, the number of iterations to convergence for POIC is around 10 iterations per pyramid level, whereas SL was trained on 10 iterations at the highest pyramid level only. Furthermore, in practice, there is a computational overhead of building the Gaussian pyramids, which increase the computation time of POIC even further. In any case, the convergence rates of POIC on the two generic face databases tested here is so poor, it negates the computational advantage of this method for this problem. Finally, SL is roughly three times more efficient than ND.

5. Conclusion

We have proposed a new approach to AAM fitting based on simulations of real fitting conditions. This approach boasts significant improvements over the fixed Jacobian, project-out and an efficient approximation of the simultaneous inverse compositional methods in both convergence accuracy and frequency. Furthermore, these improvements are afforded without sacrificing computational efficiency. Also, the method affords excellent generalizability, as evidenced by its high convergence rates.

Another advantage of this formulation is that the fitting procedure can be customized to the detector used for initialization. By building a set of training perturbations which adhere to a prior defined by the detector, better fitting performance may be achieved as the distribution of samples, and hence feature vectors, are more constrained than the uniform random perturbations used in this paper. Investigations into this aspect will be the subject of future work.

References

[1] G. Edwards, C. Taylor, T. Cootes, Interpreting Face Images Using Active Appearance Models, in: Proc. 3rd IEEE Int. Conf. on Automatic Face and Gesture Recognition (FG'98), IEEE, Nara, Japan, 1998, pp. 300–305, DOI: 10.1109/AFGR.1998.670965.

[2] T. Lehn-Schiøler, L. Hansen, J. Larsen, Mapping from Speech to Images Using Continuous State Space Models, in: Proc. First Int. Workshop on Machine Learning for Multimodal Interaction (MLMI 2004), LNCS 3361, Springer, Martigny, Switzerland, 2004, pp. 136–145.

[3] M. Stegmann, H. Larsson, Fast registration of Cardiac Perfusion MRI, in: International Society of Magnetic Resonance In Medicine, Toronto, Canada, 2003, p. 702.

[4] P. Mittrapiyanuruk, G. DeSouza, A. Kak, Accurate 3D Tracking of Rigid Objects with Occlusion Using Active Appearance Models, in: Proc. IEEE Workshop on Motion and Video Computing (WACV/MOTION 2005), Vol. 2, Breckenridge (CO), USA, 2005, pp. 90–95, DOI: 10.1109/ACVMOT.2005.15.

[5] T. Cootes, G. Edwards, C. Taylor, H. Burkhardt, B. Neuman, Active Appearance Models, in: Proc. European Conference on Computer Vision (ECCV'98), Vol. 2 of LNCS 1406, Springer, Freiburg, Germany, 1998, pp. 484–498.

[6] I. Matthews, S. Baker, Active Appearance Models Revisited, International Journal of Computer Vision 60 (2) (2004) 135–164.

[7] R. Donner, M. Reiter, G. Langs, P. Peloschek, H. Bischof, Fast Active Appearance Model Search Using Canonical Correlation Analysis, IEEE Trans. Pattern Analysis and Machine Intelligence 28 (10) (2006) 1690–1694, DOI: 10.1109/TPAMI.2006.206.

[8] X. Hou, S. Li, H. Zhang, Q. Cheng, Direct Appearance Models, in: Proc. 2001 IEEE Computer Society Conf. on Computer Vision and Pattern Recognition (CVPR 2001), Vol. 1, Kauai (HI), USA, 2001, pp. 828–833, DOI: 10.1109/CVPR.2001.990568.

[9] S. Baker, R. Gross, I. Matthews, Lucas-Kanade 20 Years On: A Unifying Framework: Part 3, Tech. rep., Robotics Institute, Carnegie Mellon University, Pittsburgh (PA), USA (2003).

[10] A. Batur, M. Hayes, Adaptive Active Appearance Models, Transactions on Image Processing 14 (11) (2005) 1707–1721, DOI: 10.1109/TIP.2005.854473.

[11] J. Saragih, R. Goecke, A Nonlinear Discriminative Approach to AAM Fitting, in: Proc. IEEE 11th Int. Conf. on Computer Vision (ICCV 2007), Rio de Janeiro, Brazil, 2007, DOI: 10.1109/ICCV.2007.4409106.

[12] S. Zhou, D. Comaniciu, Shape Regression Machine, in: Proc. 20th Int. Conf. on Information Processing in Medical Imaging (IPMI 2007), LNCS 4584, Springer, Kerkrade, The Netherlands, 2007, pp. 13–25.

[13] B. Theobald, I. Matthews, S. Baker, Evaluating Error Functions for Robust Active Appearance Models, in: Proc. 7th Int. Conf. on Automatic Face and Gesture Recognition (FG'06), Southamption, UK, 2006, pp. 149–154, DOI: 10.1109/FGR.2006.38.

[14] S. Baker, I. Matthews, Equivalence and Efficiency of Image Alignment Algorithms, in: Proc. 2001 IEEE Computer Society Conf. on Computer Vision and Pattern Recognition (CVPR 2001), Vol. 1, Kauai (HI), USA, 2001, pp. 1090–1097, DOI: 10.1109/CVPR.2001.990652.

[15] R. Gross, I. Matthews, S. Baker, Generic vs. person specific active appearance models, Image and Vision Computing 23 (12) (2005) 1080–1093, DOI: 10.1016/j.imavis.2005.07.009.

[16] X. Liu, Generic Face Alignment using Boosted Appearance Model, in: Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR 2007), Minneapolis (MN), USA, 2007, DOI: 10.1109/CVPR.2007.383265.

[17] M. Nguyen, F. De la Torre Frade, Local Minima Free Parameterized Appearance Models, in: Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR 2008), 2008, DOI: 10.1109/CVPR.2008.4587524.

[18] O. Williams, A. Blake, R. Cipolla, A sparse probabilistic learning algorithm for real-time tracking, in: Proceedings of the IEEE Ninth International Conference on Computer Vision (ICCV2003), Vol. 1, Beijing, China, 2003, pp. 353–360.

[19] S. Avidan, Support vector tracking, IEEE Transactions on Pattern Analysis and Machine Intelligence 26 (8) (2004) 1064–1072.

[20] K. Zimmermann, Fast learnable methods for object tracking, Ph.D. thesis, Czech Technical University, Prague, Czech Republic (Oct. 2008).

[21] D. Wilson, T. Martinez, The general inefficiency of batch training for gradient descent learning, Neural Networks 16 (10) (2003) 1429–1451, DOI: 10.1016/S0893-6080(03)00138-2.

[22] L. Bottou, Stochastic Learning, in: Advanced Lectures on Machine Learning, LNAI 3176, Springer, 2004, pp. 146–168.

[23] J. Kivinen, A. Smola, R. Williamson, Online Learning with Kernels, in: Advances in Neural Information Processing Systems 14 (NIPS 2001), MIT Press, Vancouver, Canada, 2002, pp. 785–793.

[24] J. Spall, Stochastic Optimization: Stochastic Approximation and Simulated Annealing, in: J. Webster (Ed.), Encyclopedia of Electrical and Electronics Engineering, Vol. 20, Wiley, New York, USA, 1998, pp. 529–542.

[25] J. Spall, Multivariate stochastic approximation using a simultaneous perturbation gradient approximation, IEEE Trans. Automatic Control 37 (3) (1992) 332–341, DOI: 10.1109/9.119632.

[26] J. Spall, Implementation of the Simultaneous Perturbation Algorithm for Stochastic Optimization, IEEE Trans. Aerospace and Electronic Systems 34 (3) (1998) 817–823, DOI: 10.1109/7.705889.

[27] T. Cootes, G. Edwards, C. Taylor, A Comparative Evaluation of Active Appearance Model Algorithms, in: Proc. British Machine Vision Conference (BMVC'98), Southampton, UK, 1998, pp. 680–689.

[28] J. Saragih, R. Goecke, Iterative Error Bound Minimisation for AAM Alignment, in: Proc. 18th Int. Conf. on Pattern Recognition (ICPR 2006), Vol. 2, Hong Kong, 2006, pp. 1192–1195, DOI: 10.1109/ICPR.2006.730.

[29] D. Liu, J. Nocedal, On the Limited Memory BFGS Method for Large Scale Optimization, Mathematical Programming 45 (1–3) (1989) 503–528.

[30] M. Nordstrøm, M. Larsen, J. Sierakowski, M. Stegmann, The IMM Face Database - An Annotated Dataset of 240 Face Images, Tech. rep., Informatics and Mathematical Modelling, Technical University of Denmark (May 2004).

[31] K. Messer, J. Matas, J. Kittler, J. Lüttin, G. Maitre, XM2VTSDB: The Extended M2VTS Database, in: Proc. 2nd Int. Conf. Audio- and Video-Based Biometric Person Authentication (AVBPA'99), Washington (DC), USA, 1999, pp. 72–77.

[32] `http://www.isbe.man.ac.uk/~bim/data/xm2vts/xm2vts_markup.html`.