4

SHAPE AND TEXTURE-BASED DEFORMABLE MODELS FOR FACIAL IMAGE ANALYSIS

Stan Z. Li and Zhen Lei

Institute of Automation, Chinese Academy of Sciences, Beijing, China

Ying Zheng and Zeng-Fu Wang

Department of Automation, Institute of Information Science and Technology, University of Science and Technology of China, Hefei, China

In this chapter we introduce concepts and algorithms of shape- and texture-based deformable models — more specifically Active Shape Models (ASMs), Active Appearance Models (AAMs), and Morphable Models — for facial image analysis. Such models, learned from training examples, allow admissible deformations under statistical constraints on the shape and/or texture of the pattern of interests. As such, the deformation is in accordance with the specific constraints on the pattern. Based on analysis of problems with the standard ASM and AAM, we further describe enhanced models and algorithms, namely Direct Appearance Models (DAMs) and a Texture-Constrained ASM (TC-ASM), for improved fitting of shapes and textures. A method is also described for evaluation of goodness of fit using an ASM. Experimental results are provided to compare different methods.

1. INTRODUCTION

Many image based systems require alignment between an object in the input image and a target object. The alignment quality can have a great impact on system

Address all correspondence to: Stan Z. Li, Center for Biometrics and Security Research, National Laboratory of Pattern Recognition, Institute of Automation, Chinese Academy of Sciences, 95 Zhongguancun Donglu, Beijing 100080, China. Phone: +86-10-8262 6787; Fax: +86- 10-6265 9350. szli@cbsr.ia.ac.cn, szli@nlpr.ia.ac.cn. Web: http://www.cbsr.ia.ac.cn/, http://www.cbsr.ia.ac.cn/homapage/szli.

performance. For face analysis, in particular, both shapes and textures provide important clues useful for characterizing faces. The task of face alignment is to accurately locate facial features such as the eyes, nose, mouth, and outline, and to normalize facial shape and texture. Accurate extraction and alignment of these features offers advantages for many applications.

A sort of the most successful face alignment method is the deformable model, which can represent variations in either the shape or texture of the target objects. As two typical deformable model types, the active shape models (ASMs) [1] and active appearance models (AAMs) [2, 3] have been widely used as alignment algorithms in medical image analysis and face analysis [4] for the past decade. The standard ASM consists of two statistical models: (1) a global shape model, which is derived from the landmarks in the object contour, and (2) a local appearance model, which is derived from the profiles perpendicular to the object contour around each landmark. The ASM uses local models to find the candidate shape and the global model to constrain the searched shape. The AAM makes use of subspace analysis techniques, PCA in particular, to model both shape variation and texture variation, and the correlations between them. The integrated shape and texture is referred to as *appearance*. In searching for a solution, it assumes linear relationships between appearance variation and texture variation, and between position variation and texture variation; and learns the two linear regression models from training data. Minimization in high-dimensional space is reduced in the two models. This strategy is also developed in the active blob model[5].

ASMs and AAMs can be expanded in several ways. The concept, originally proposed for the standard frontal view, can be extended to multi-view faces, either by using piecewise linear modeling [6] or nonlinear modeling [7]. Cootes and Taylor show that imposing constraints such as fixing eye locations can improve AAM search results [8]. Blanz and Vetter extended morphable models and the AAM to model the relationship of 3D head geometry and facial appearance [9]. Li et al. [10] present a method for learning 3D face shape modeling from 2D images based on a shape-and-pose-free texture model. In Duta et al. [11], the shapes are automatically aligned using procrustean analysis, and clustered to obtain cluster prototypes and statistical information about intra-cluster shape variation. In Ginneken et al. [12], a K-nearest-neighbors classifier is used and a set of features selected for each landmark to build local models. Baker and colleagues [13] propose an efficient method called an "inverse compositional algorithm" for alignment. Ahlberg [14] extends the AAM to a parametric method called an Active Appearance algorithm to extract positions parameterized by 3D rotation, 2D translation, scale, and six Action Units (controlling the mouth and the eyebrows). In the direct appearance model (DAM) [15, 16], shape is modeled as a linear function of texture. Using such an assumption, Yan et al. [17] propose a texture-constrained ASM (TC-ASM), which has the advantage of an ASM in having good localization accuracy and that of an AAM in having insensitivity to initialization. To construct an effective evaluation function, a statistical learning approach was

proposed for face alignment by Huang et al. [18] using a nonlinear classification function learned from a training set of positive and negative training examples.

The following sections first describe the classical ASM and AAM. We will then briefly review the 3D Morphable Model as an important 3D deformable model. After that, two of the improved face alignment algorithms — DAM and TC-ASM — will be introduced based on the analysis of the problems of classical the ASM and AAM. Then an alignment quality evaluation mechanism is addressed before the experimental results and conclusion to end this chapter are presented.

For all the algorithms presented here, a training set of shape–texture pairs is assumed to be available and denoted as $\mathbf{\Omega} = \{(S_0, T_0)\}$, where a *shape* $S_0 = ((x_1, y_1), \ldots, (x_K, y_K)) \in \mathbb{R}^{2K}$ is a sequence of K points in the 2D image plane, and a *texture* T_0 is the patch of pixel intensities enclosed by S_0. Let $\overline{S}$ be the mean shape of all the training shapes, as illustrated in Figure 1. All the shapes are aligned or warping to the tangent space of the mean shape $\overline{S}$. After that texture T_0 is warped correspondingly to $T \in \mathbb{R}^L$, where L is the number of pixels in the mean shape $\overline{S}$. The warping may be done by pixel value interpolation, e.g., using a triangulation or thin plate spline method.

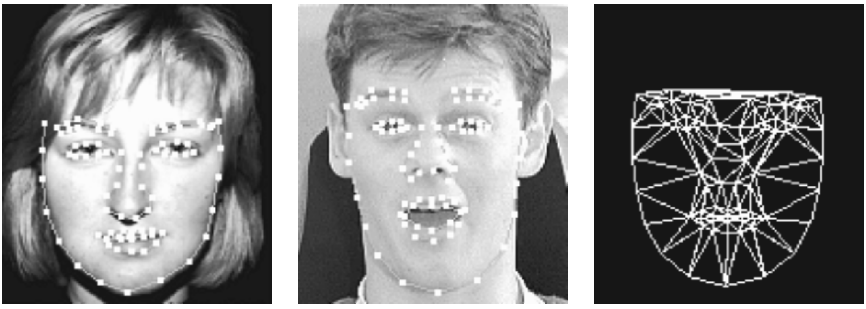

Figure 1. Two face instances labeled with 83 landmarks and the mesh of the mean shape. Reprinted with permission from SC Yan, C Liu, SZ Li, HJ Zhang, H Shum, QS Cheng. 2003. Face alignment using texture-constrained active shape models. *Image Vision Comput* **21**(1):69–75. Copyright ©2003, Elsevier. See attached CD for color version.

2. CLASSICAL DEFORMABLE MODELS

There are two classical deformable models for 2D face analysis — the Active Shape Model (ASM) and the Active Appearance Model (AAM). We first look through them; the model for 3D face analysis will be addressed later.

The ASM seeks to match a set of model points to an image by searching along profiles of each point under the constraint of a statistical shape model. The AAM

seeks to match both the position of the model points and a representation of the texture to an image by updating the model parameters using the difference between the current synthesized image and the target image.

There are three key differences between the two models [19]:

1. The ASM only uses models of the image texture in small regions about each landmark point, whereas the AAM uses a model of the appearance of the whole of the region (usually inside a convex hull around the points).

2. The ASM searches around the current position, typically along profiles normal to the boundary, whereas the AAM only samples the image enclosed by the current position.

3. The ASM essentially seeks to minimize the distance between model points and the corresponding points found in the image, whereas the AAM seeks to minimize the difference between the synthesized model image and the target image.

2.1. Active Shape Model

In the ASM a shape is represented as a vector s in the low-dimensional shape eigenspace $\mathbb{R}^k$, spanned by $k\,(<2K)$ principal modes (major eigenvectors) learned from the training shapes. A shape S could be linearly obtained from the shape eigenspace

$$S = \overline{S} + \mathbf{U}s, \tag{1}$$

where $\mathbf{U}$ is the matrix consisting of k principal modes of the covariance of $\{S_0\}$.

The local appearance models, which describe a local image feature around each landmark, are modeled as the first derivatives of the sampled profiles perpendicular to the landmark contour [4]. For the jth landmark ($j=1,\cdots,K$), we can derive the mean profile $\overline{g}_j$ and the covariance matrix $\mathbf{\Sigma}_j^g$ from the jth profile examples directly. At the current position $(x_j^{(n-1)}, y_j^{(n-1)})$ of the jth landmark, the local appearance models find the "best" candidate, (x_j^n, y_j^n), in neighborhood $N(x_j^{(n-1)}, y_j^{(n-1)})$ surrounding $(x_j^{(n-1)}, y_j^{(n-1)})$, by minimizing the energy:

$$(x_j^n, y_j^n) = \underset{(x,y)\in N(x_j^{(n-1)}, y_j^{(n-1)})}{\arg\min} \|g_j(x,y) - \overline{g}_j\|^2_{\mathbf{\Sigma}_j^g}, \tag{2}$$

where $g_j(x,y)$ is the profile of the jth landmark at (x,y) and $\|X\|^2_{\mathbf{A}} = X^T\mathbf{A}^{-1}X$ is the Mahalanobis distance measure with respect to a real symmetric matrix $\mathbf{A}$.

After relocating all the landmarks using the local appearance models, we obtain a new candidate shape S_{lm}^n. The solution in shape eigenspace is derived by maximizing the likelihood:

$$s^n = \arg\max_s p(S^n_{lm}|s) = \arg\min_s Eng(S^n_{lm}; s), \tag{3}$$

where[1]

$$Eng(S^n_{lm}; s) = \lambda \|S^n_{lm} - S^n_{lm}{}'\|^2 + \|s^n_{lm} - s\|^2_{\mathbf{\Lambda}}. \tag{4}$$

In above equation, $s^n_{lm} = U^T(S^n_{lm} - \overline{S})$ is the projection of S^n_{lm} to the shape eigenspace, $S^n_{lm}{}' = \overline{S} + U s^n_{lm}$ is the reconstructed shape, and $\mathbf{\Lambda}$ is the diagonal matrix of the largest eigenvalues of the training data $\{S_i\}$. The first term is the squared Euclidean distance from S^n_{lm} to the shape eigenspace, and the second is the squared Mahalanobis distance between s^n_{lm} and s; λ balances the two terms.

Using the local appearance models leads to fast convergence to the local image evidence. However, since they are modeled based on the local features, and the "best" candidate point is only evaluated in the local neighborhood, the solution of the ASM is often suboptimal, dependent on the initialization.

2.2. Active Appearance Model

In the AAM, a shape is also modeled by k ($<2K$) principal modes learned from the training shapes by PCA, just as Eq. (1) shows with the ASM.

After aligning each training shape S_0 to the mean shape and warping the corresponding texture T_0 to T, the warped textures are aligned to the tangent space of the mean texture $\overline{T}$ by using an iterative approach [2]. The PCA model for the warped texture is obtained as

$$T = \overline{T} + \mathbf{V}t, \tag{5}$$

where $\mathbf{V}$ is the matrix consisting of ℓ principal orthogonal modes of variation in $\{T\}$, and t is the vector of texture parameters. The projection from T to t is

$$t = \mathbf{V}^T(T - \overline{T}) = \mathbf{V}^T T. \tag{6}$$

By this, the L pixel values in the mean shape is represented as a point in the texture subspace $\mathbb{S}_t$ in $\mathbb{R}^\ell$.

The appearance of each example is a concatenated vector:

$$A = \begin{pmatrix} \mathbf{\Lambda} s \\ t \end{pmatrix}, \tag{7}$$

where $\mathbf{\Lambda}$ is a diagonal matrix of weights for the shape parameters allowing for the difference in units between the shape and texture variation, typically defined as $r\mathbf{I}$. Again, by applying PCA on the set $\{A\}$, one gets

$$A = \mathbf{W}a, \tag{8}$$

where $\mathbf{W}$ is the matrix consisting of principal orthogonal modes of the variation in $\{A\}$ for all training samples. The appearance subspace $\mathbb{S}_a$ is modeled by

$$a = \mathbf{W}^T A. \tag{9}$$

The search for an AAM solution is guided by the following difference between the texture T_{im} in the image patch and the texture T_a reconstructed from the current appearance parameters:

$$\delta T = T_{im} - T_a. \tag{10}$$

More specifically, the search for a face in an image is guided by minimizing the norm $\|\delta T\|$. The AAM assumes that the appearance displacement δa and the position (including coordinates (x, y), scale s, and rotation parameter θ) displacement δp are linearly correlated to δT:

$$\delta a = \mathbf{A}_a \delta T \tag{11}$$

$$\delta p = \mathbf{A}_p \delta T \tag{12}$$

The prediction matrices $\mathbf{A}_a, \mathbf{A}_p$ are to be learned from the training data by using linear regression. In order to estimate $\mathbf{A}_a$, a is displaced systematically to induce $(\delta a, \delta T)$ pairs for each training image. Due to the large consumption of memory required for the learning of $\mathbf{A}_a$ and $\mathbf{A}_p$, learning has to be done with a small, limited set of $\{\delta a, \delta T\}$.

2.3. 3D Morphable Model

While the ASM and AAM are for 2D image pattern analysis, in this section we temporarily deviate from analysis of a 2D face, and extend the dimension of face data to 3D by introducing morphable models [9, 20, 21]. With the presence of convenient 3D acquiring equipment and the development of the computer hardware, 3D face analysis has now become feasible and promising since it is invariant to the influence of pose, illumination, and expression. One of the most crucial problems for all 3D data-processing systems is the alignment between the input data and the standard. The 3D alignment may involve many rigid or non-rigid transformations. For 3D face analysis, in particular, alignment means reconstruction of a normalized 3D face model from either input 2D face images or unrestrained 3D data. The 3D Morphable Model (3DMM), as a typical 3D deformable model, inherits both the spirit of a multidimensional morphable model [22] and the AAM.

The 3DMM is a model of faces represented in 3D where shape information is separated from texture information. The shape and texture models are learned from a database of 3D faces, i.e., faces acquired by a 3D *Cyberware*^{TM} scanner. Building a 3DMM requires transforming the shape and texture spaces into

vector spaces for which any convex combination of exemplar shapes and textures describes a realistic human face. *Correspondence* is the basic requirement for constructing such a vector space. In [23], correspondences are established between all exemplar faces and a reference face by an optical flow algorithm. This scheme brings a consistent labeling of vertices and corresponding albedos across the whole set of exemplar faces. The shape of an exemplar face is then represented by a shape vector $S^{ex} = ((x_1, y_1, z_1) \ldots, (x_K, y_K, z_K)) \in \mathbb{R}^{3K}$ that contains the x, y, z coordinates of K vertices. The texture of the face is represented by a texture vector $T^{ex} = ((R_1, G_1, B_1) \ldots, (R_K, G_K, B_K)) \in \mathbb{R}^{3K}$ that contains the R, G, B texture values sampled at the same K vertices.

A new face can then be generated by convex combination of the K exemplar faces, with their shape and texture vectors, S and T, expressed as

$$S = \sum_{i=1}^{K} a_i S_i^{ex} \quad T = \sum_{i=1}^{K} b_i T_i^{ex} \quad \sum_{i=1}^{K} a_i = \sum_{i=1}^{K} b_i = 1. \tag{13}$$

Again, PCA is applied separately on the shape and texture space to reduce dimensionality. Now, instead of describing a new face as a convex combination of exemplars, as in Eq. (13), we use the similar shape and texture PCA model of Eq. (1), (5) as

$$S = \overline{S} + \mathbf{U}s \quad T = \overline{T} + \mathbf{V}t. \tag{14}$$

Note that U and V are the matrices consisting of orthogonal modes of variations in $\{S^{ex}\}$ and $\{T^{ex}\}$. The 3DMM shape and texture coefficient vectors s and t are low-dimensional coding of the identity of a face invariant to pose and illumination influence. Given an input 2D image under arbitrary pose and illumination conditions or unrestrained 3D face data, the 3DMM can recover the vectors of s and t by an *analysis by synthesis*, providing an alignment between input face and exemplar faces in the database.

3. MOTIVATIONS FOR IMPROVEMENTS

ASM uses the local appearance models to search along the profiles of candidate points. It leads to fast convergence to the local image evidence. However, since they are modeled based on local features, and the "best" candidate point is only evaluated in the local neighborhood, the solution of the ASM is often suboptimal, dependent on the initialization.

By analyzing the relationships between the shape, texture, and appearance subspaces in the AAM, we will show the defects of the model. Thereby, we suggest a property that an ideal appearance model should have, which motivates us to propose improvements to the classical model.

First, let us look into the relationship between shape and texture from an intuitive viewpoint. A texture (i.e., the patch of intensities) is enclosed by a shape

(before aligning to the mean shape); the same shape can enclose different textures (i.e., configurations of pixel values). However, the reverse is not true: different shapes cannot enclose the same texture. So the mapping from the texture space to the shape space is many-to-one. The shape parameters should be determined completely by texture parameters but not vice versa.

Let us now look further into the correlations or constraints between the linear subspaces $\mathbb{S}_s, \mathbb{S}_t$ and $\mathbb{S}_a$ in terms of their dimensionalities or ranks. Let us denote the rank of space $\mathbb{S}$ by dim($\mathbb{S}$). We have the following analysis:

1. When dim($\mathbb{S}_a$)=dim($\mathbb{S}_t$)+dim($\mathbb{S}_s$), the shape and texture parameters are independent of each other, and there exist no mutual constraints between the s and t parameters.

2. When dim($\mathbb{S}_t$)<dim($\mathbb{S}_a$)<dim($\mathbb{S}_t$)+ dim($\mathbb{S}_s$), not all the shape parameters are independent of the texture parameters. That is, one shape can correspond to more than one texture configuration in it, which conforms our intuition.

3. One can also derive the relationship dim($\mathbb{S}_t$)<dim($\mathbb{S}_a$) from Eqs. (7) and (8) and write
$$\mathbf{W}a = \begin{pmatrix} \mathbf{\Lambda}s \\ t \end{pmatrix} \tag{15}$$
when that s contains some components that are independent of t.

4. However, in the AAM it is often the case where dim($\mathbb{S}_a$)<dim($\mathbb{S}_t$) if the dimensionalities of $\mathbb{S}_a$ and $\mathbb{S}_t$ are chosen to retain, say, 98% of the total variations, which is reported by Cootes [2] and also observed by us. The consequence is that some admissible texture configurations cannot be seen in the appearance subspace because dim($\mathbb{S}_a$)<dim($\mathbb{S}_t$), and therefore cannot be reached by the AAM search. We consider this a flaw in the AAM's modeling of its appearance subspace.

From the above analysis we conclude that the ideal model should be dim($\mathbb{S}_a$) = dim($\mathbb{S}_t$), and hence that s is completely linearly determinable by t. In other words, the shape should be linearly dependent on the texture, so that dim($\mathbb{S}_t \cup \mathbb{S}_s$) = dim($\mathbb{S}_t$). The direct appearance model (DAM) is proposed mainly for this purpose.

Another motivation of the DAM is memory consumption: the regression of $\mathbf{A}_a$ with the AAM is very memory consuming. The AAM prediction needs to model linear the relationship between appearance and the texture difference according to Eq. (11). However, both δa and δT are high-dimensional vectors, and therefore the storage size of training data generated for learning Eq. (11) increases very rapidly as the dimensions increase. It is very difficult to train the AAM for $\mathbf{A}_a$ even with a moderate number of images. Learning in a low-dimensional space will relieve the burden.

4. DIRECT APPEARANCE MODELS

In this section we introduce an improved appearance model, called the Direct Appearance Model (DAM), for aligning and estimating face appearances.

The new appearance model is motivated by our findings of a flaw in AAM modeling and the difficulties in training the AAM presented in previous section. The DAM model overcomes these problems by its proper subspace modeling based on the fundament that the mapping from the texture subspace to the shape subspace is many-to-one, and therefore a shape can be determined entirely by the texture in it. From these relationships, the DAM model considers an appearance that is composed of both shape and texture, to be determinable by using just the corresponding texture. The DAM uses the texture information *directly* to predict the shape and to update the estimates of position and appearance (hence the name DAM) in contrast to the AAM's crucial idea of modeling the AAM appearance subspace from shape and texture combined. In this way, the DAM includes admissible textures previously unseen by the AAM and improves convergence and accuracy.

Another merit of the DAM is that it predicts the new face position and appearance based on principal components of texture difference vectors, instead of the raw vectors themselves as with the AAM. This cuts down the memory requirement to a large extent and further improves convergence and accuracy. The claimed advantages of the DAM are substantiated by comparative experimental results in Section 7.1.

4.1. DAM Modeling and Training

DAM consists of a shape model, a texture model, and a prediction model. It predicts the shape parameters directly from the texture parameters. The shape and texture models are built based on PCA in the same way as with the AAM. The prediction model includes two parts: prediction of position and prediction of texture.

Recall the conclusions we made earlier: (1) an ideal model should have $\dim(\mathbb{S}_a) = \dim(\mathbb{S}_t)$, and (2) shape should be computable uniquely from texture but not vice versa. We propose the following prediction model by assuming a linear relationship between shape and texture:

$$s = \mathbf{R}t + \varepsilon, \tag{16}$$

where $\varepsilon = s - \mathbf{R}t$ is noise and $\mathbf{R}$ is a $k \times l$ projection matrix. Denoting the expectation by $E(\cdot)$, if all the elements in variance matrix $E(\varepsilon\varepsilon^T)$ are small enough, the linear assumption made in Eq. (16) is approximately correct. This is true, as will be verified later by experiments. Define the objective cost function

$$C(\mathbf{R}) = E(\varepsilon^T\varepsilon) = \text{trace}[E(\varepsilon\varepsilon^T)]. \tag{17}$$

$\mathbf{R}$ is learned from training example pairs $\{(s, t)\}$ to minimize the cost function.

Consider variation $\delta C(\mathbf{R})$ caused by $\delta\mathbf{R}$:

$$\begin{aligned}
&\delta C(\mathbf{R}) \qquad (18)\\
&= \text{trace}\{E([s-(\mathbf{R}+\delta\mathbf{R})t][s-(\mathbf{R}+\delta\mathbf{R})t]^T)\}\\
&\quad -\text{trace}[E\{[s-\mathbf{R}t][s-\mathbf{R}t]^T\}]\\
&= \text{trace}\{E[\mathbf{R}tt^T\delta\mathbf{R}^T+\delta\mathbf{R}tt^T\mathbf{R}\\
&\quad -st^T\delta\mathbf{R}^T-\delta\mathbf{R}ts^T]\}\\
&= \text{trace}\{\mathbf{R}E(tt^T)\delta\mathbf{R}^T+\Delta\mathbf{R}E(tt^T)\mathbf{R}\\
&\quad -E(st^T)\Delta\mathbf{R}^T-\delta\mathbf{R}E(ts^T)\}.
\end{aligned}$$

Letting $\delta C(\mathbf{R})=0$, we get

$$\begin{aligned}
&\text{trace}\{\delta\mathbf{R}E(tt^T)\delta\mathbf{R}^T+\delta\mathbf{R}E(tt^T)\mathbf{R}\} \qquad (19)\\
&\quad = \text{trace}\{E(st^T)\Delta\mathbf{R}^T+\Delta\mathbf{R}E(ts^T)\}
\end{aligned}$$

for any $\|\delta\mathbf{R}\|\to 0$. Substituting $\delta\mathbf{R}$ by $\epsilon 1_{i,j}$ for any (i,j), where $\epsilon\to 0$ and $1_{i,j}$ is the matrix in which entry (i,j) is 1 and 0 elsewhere, we arrive at $\mathbf{R}E(tt^T)=E(st^T)$, and hence obtain an optimal solution:

$$\mathbf{R}=E(st^T)[E(tt^T)]^{-1}. \qquad (20)$$

The minimized cost is the trace of the following:

$$E(\varepsilon\varepsilon^T)=E(ss^T)-\mathbf{R}E(tt^T)\mathbf{R}^T. \qquad (21)$$

Instead of using δT directly as in the AAM search (cf. Eq. (12)), we use principal components of it, $\delta T'$, to predict the position displacement:

$$\delta p=\mathbf{R}_p\delta T', \qquad (22)$$

where $\mathbf{R}_p$ is the prediction matrix learned by using linear regression. To do this, we collect texture differences induced by small position displacements in each training image, and perform PCA on this data to get the projection matrix $\mathbf{H}^T$. A texture difference is projected onto this subspace as

$$\delta T'=\mathbf{H}^T\delta T, \qquad (23)$$

where $\delta T'$ is about 1/4 of δT in dimensionality, and this makes the prediction more stable. The DAM regression in Eq. (22) requires much less memory than the AAM regression in Eq. (11). This is because p is of much lower dimension than a and $\delta T'$ much lower than δT. This will be illustrated by numbers later.

Assume that a training set is given as $\mathbf{A} = \{(S_i, , T_i)\}$ where a shape $S_i = ((x_1^i, y_1^i), \ldots, (x_K^i, y_K^i)) \in \mathbb{R}^{2K}$ is a sequence of K points in the 2D image plane, and a texture T_i is the patch of image pixels enclosed by S_i. The DAM learning consists of two parts: (1) learning $\mathbf{R}$, and (2) learning $\mathbf{H}$ and $\mathbf{R}_p$. (1) $\mathbf{R}$ is learned from the shape–texture pairs $\{s, t\}$ obtained from the landmarked images. (2) To learn $\mathbf{H}$ and $\mathbf{R}_p$, artificial training data are generated by perturbing the position parameters p around the landmark points to obtain $\{\delta p, \delta T\}$; then learn $\mathbf{H}$ from $\{\delta T\}$ using PCA; $\delta T'$ is computed after that; and, finally, $\mathbf{R}_p$ is derived from $\{\delta p, \delta T'\}$.

The DAM regression in Eq. (22) requires much less memory than the AAM regression in Eq. (11); typically, a DAM needs only about 1/20th the memory required by an AAM. For the DAM, there are 200 training images, 4 parameters for the position $(x, y, \theta, scale)$, and 6 disturbances for each parameter to generate training data for the training $\mathbf{R}_p$. So the size of the training data for a DAM is $200 \times 4 \times 6 = 4,800$. For AAM there are 200 training images, 113 appearance parameters, and 4 disturbances for each parameter to generate training data for training $\mathbf{A}_a$. The size of the training data set for $\mathbf{A}_a$ is $200 \times 113 \times 4 = 90,400$. Therefore, the size of the training data set for an AAM's prediction matrices is $90,400 + 4,800 = 95,200$, which is 19.83 times that for a DAM. On a PC, for example, the memory capacity for AAM training with 200 images would allow DAM training with 3,966 images.

Note that there is a variant of a basic AAM [4], which uses texture difference to predict shape difference. The prediction of shape is done by $\delta s = \mathbf{B}\delta T$. However, this variant is not as good as the basic AAM [4].

4.2. DAM Search

The DAM prediction model leads to the following search procedure. The DAM search starts with the mean shape and mean texture, equivalent to the mean appearance with $a_0 = 0$, at a given initial position p_0. The texture difference δT is computed from the current shape patch at the current position, and its principal components are used to predict and update p and s using the DAM linear models described above. If $\|\delta T\|$ calculated using the new appearance at the position is smaller than the old one, the new appearance and position are accepted; otherwise, the position and appearance are updated by amounts $\kappa\delta a$ and $\kappa\delta p$ with varying κ values. The search algorithm is summarized below:

1. Initialize position parameters p_0, and set shape parameters $s_0 = 0$;

2. Get texture T_{im} from the current position, project it into the texture subspace $\mathbb{S}_t$ as t; reconstruct the texture T_{rec}, and compute texture difference $\delta T_0 = T_{im} - T_{rec}$ and the energy $E_0 = \|\delta T_0\|^2$;

3. Compute $\delta T' = \mathbf{H}^T \delta T$, and get the position displacement $\delta p = \mathbf{R}_p \delta T'$;

4. Set step size $\kappa = 1$;

5. Update $p = p_0 - \kappa\delta p$, $s = \mathbf{R}t$;

6. Compute difference texture δT using the new shape at the new position, and its energy $E_0 = \|\delta T_0\|^2$;

7. If $|E - E_0| < \epsilon$, the algorithm is converged; exit;

8. If $E < E_0$, then let $p_0 = p$, $s_0 = s$, $\delta T_0 = \delta T$, $E_0 = E$, goto 3;

9. Change κ to the next smaller number in $\{1.5, 0.5, 0.25, 0.125, \ldots, \}$, goto 5;

The above DAM search can be performed with a multi-resolution pyramid structure to improve the result.

4.3. Multi-View DAM

In multi-view face alignment, the whole range of views from frontal to side views are partitioned into several sub-ranges, and one DAM model is trained to represent the shape and texture for each sub-range. Which view DAM model to use may be decided by using some pose estimate for static images. In the case of face alignment from video, the previous view plus the two neighboring view DAM models may be attempted, and then the final result is chosen to be the one with the minimum texture residual error.

The full range of face poses are divided into 5 view sub-ranges: $[-90°, -55°]$, $[-55°, -15°]$, $[-15°, 15°]$, $[15°, 55°]$, and $[55°, 90°]$, with $0°$ being the frontal view. The landmarks for frontal, half-side, and full-side view faces are illustrated in Figure 2. The dimensions of shape and texture vectors before and after the PCA dimension reductions are shown in Table 1, where the dimensions after PCA are chosen to be such that 98% of the corresponding total energies are retained. The texture appearances due to respective variations in the first three principal components of texture are demonstrated in Figure 3.

The left- models and right-side models are reflections of each other, andn thus we only need train one side. So we train $[-15°, 15°]$, $[15°, 55°]$, and $[55°, 90°]$ for the 5 models. We can find the corresponding model for all the face with a view in $[-90°, 90°]$.

The multi-view DAM search has a similar process to that of the standard DAM. The difference lies in the beginning of the iteration where multi- view DAM has to determine which view the input image belongs to and select a proper DAM model. Note that p can be computed from δT in one step as $\delta p = \mathbf{R}_T \delta T$, where $\mathbf{R}_T = \mathbf{R}_p \mathbf{H}^T$, instead of two steps as in Eqs. (22) and (23). The search algorithm is summarized below:

Table 1. Dimensionalities of shape and texture variations for face data

View	#1	#2	#3	#4	#5
Fontal	87	69	3185	144	878
Half-Side	65	42	3155	144	1108
Full-Side	38	38	2589	109	266

#1 Number of landmark points. #2 Dimension of shape space $\mathbb{S}_s$. #3 Number of pixel points in the mean shape. #4 Dimension of texture space $\mathbb{S}_t$. #5 Dimension of texture variation space ($\delta T'$). Reprinted with permission from SZ Li, SC Yan, HJ Zhang, QS Cheng. 2002. Multi-view face alignment using direct appearance models. *Proc 5th Int Conf Automatic Face Gesture Recogn*, pp. 309–314. Copyright ©2002, IEEE.

1. Initialize position parameters p_0, and determine the view by which to select the DAM model to use; set shape parameters $s_0 = 0$;
2. Get texture T_{im} from the current position; project it into texture subspace $\mathbb{S}_t$ as t; reconstruct texture T_a, and compute texture difference $\delta T_0 = T_{im} - T_a$ and the energy $E_0 = \|\delta T_0\|^2$;
3. get position displacement $\delta p = \mathbf{R}_T \delta T$;
4. Set step size $\kappa = 1$;
5. Update $p = p_0 - \kappa \delta p$, $s = \mathbf{R}t$;
6. Compute difference texture δT using the new shape at the new position, and its energy $E = \|\delta T\|^2$;
7. If $|E - E_0| < \epsilon$, the algorithm is converged; exit;
8. If $E < E_0$, then let $p_0 = p$, $s_0 = s$, $\delta T_0 = \delta T$, $E_0 = E$, goto 3;
9. Change κ to the next number in $\{1.5, 0.5, 0.25, 0.125, \ldots, \}$, goto 5.

In our implementation, the initialization and pose estimation are performed automatically by using a robust real-time multi-view face detector [24], as shown in Figure 4. A multi-resolution pyramid structure is used in the search to improve the result. Figure 5 demonstrates scenarios of how DAM converges.

When the face has undergone large variation due to stretch in either the x or y direction, model fitting can be improved by allowing different scales in the two directions. This is done by splitting the scale parameter in two: s_x and s_y. The improvement is demonstrated in Figure 6.

Figure 2. Frontal, half-side, and full-side view faces and the labeled landmark points. Reprinted with permission from SZ Li, SC Yan, HJ Zhang, QS Cheng. 2002. Multi-view face alignment using direct appearance models. *Proc 5th Int Conf Automatic Face Gesture Recogn*, pp. 309– 314. Copyright ©2002, IEEE.

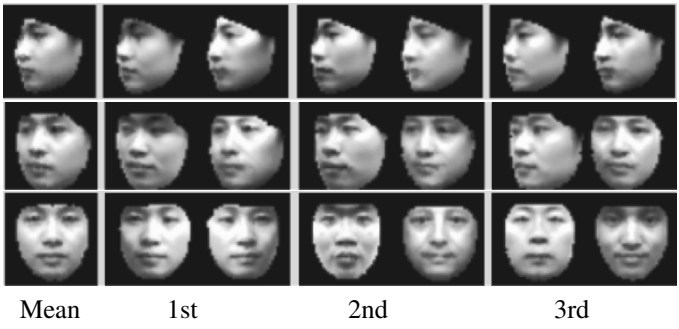

Figure 3. Texture and shape variations due to variations in the first three principal components of the texture (the shapes change in accordance with $s = \mathbf{R}t$) for full-side ($\pm 1\sigma$), half-side ($\pm 2\sigma$), and frontal ($\pm 3\sigma$) views. Reprinted with permission from SZ Li, SC Yan, HJ Zhang, QS Cheng. 2002. Multi-view face alignment using direct appearance models. *Proc 5th Int Conf Automatic Face Gesture Recogn*, pp. 309–314. Copyright ©2002, IEEE.

5. TEXTURE-CONSTRAINED ACTIVE SHAPE MODEL

A TC-ASM [17] imposes the linear relationship of the direct appearance model (DAM) to improve the ASM search. The motivation is as follows. The ASM has better accuracy in shape localization than the AAM when the initial shape is placed close enough to the true shape, whereas the latter model incorporates information about texture enclosed in the shape and hence yields lower texture reconstruction error. However, the ASM makes use of constraints near the shape only, without a global optimality criterion, and therefore the solution is sensitive to the initial shape position. In the AAM, the solution-finding process is based on the linear relationship between the variation of the position and the texture reconstruct error. The reconstruct error, δT, is influenced very much by the illumination. Since δT

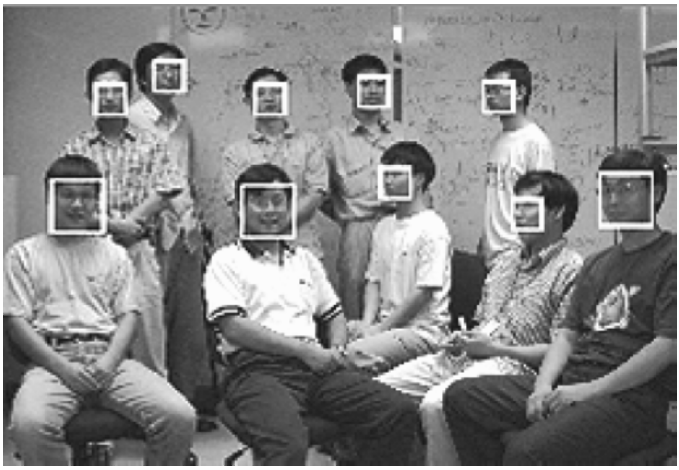

Figure 4. Initial alignment provided by a multi-view face detector. Reprinted with permission from SZ Li, SC Yan, HJ Zhang, QS Cheng. 2002. Multi-view face alignment using direct appearance models. *Proc 5th Int Conf Automatic Face Gesture Recogn*, pp. 309–314. Copyright ©2002, IEEE.

Figure 5. DAM aligned faces (from left to right) at the 0th, 5th, 10th, and 15th iterations, and the original images for (top–bottom) frontal, half-side and full-side view faces. Reprinted with permission from SZ Li, SC Yan, HJ Zhang, QS Cheng. 2002. Multi-view face alignment using direct appearance models. *Proc 5th Int Conf Automatic Face Gesture Recogn*, pp. 309–314. Copyright ©2002, IEEE.

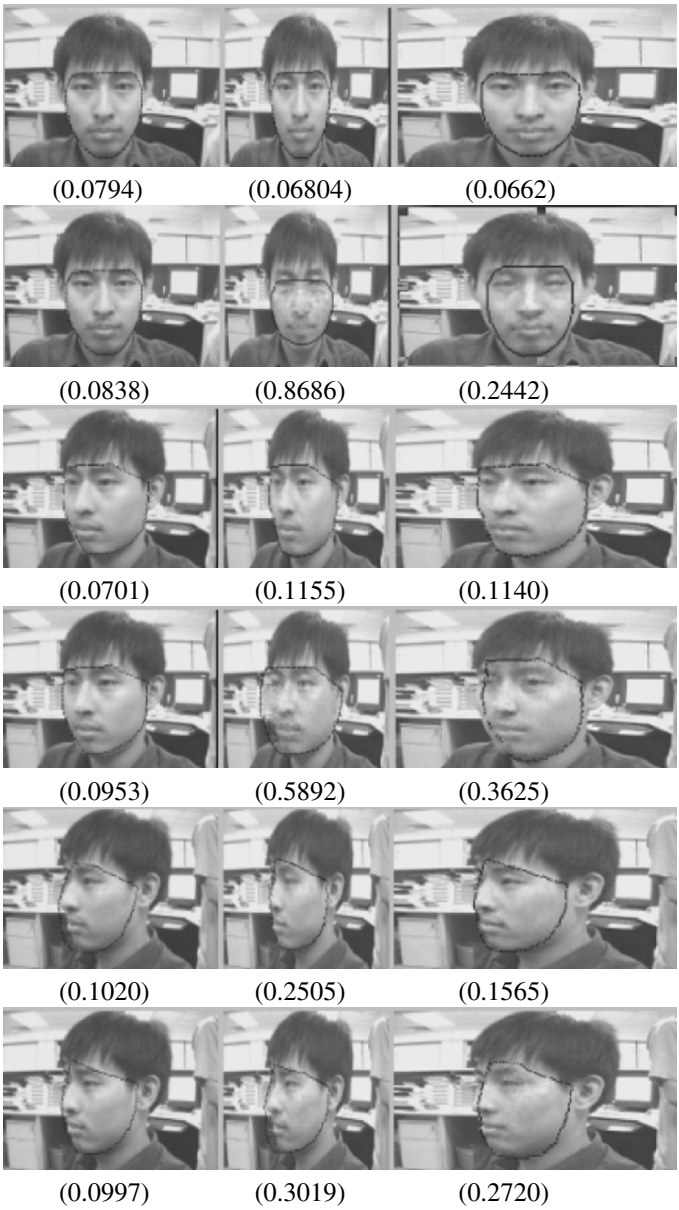

Figure 6. Results of non-isometric (top of each of the three blocks) and isometric (bottom) search for frontal (top block), half-side (middle block), and full-side (bottom block) view faces. From left to right of each row are normal, and stretched faces. The number below each result is the corresponding residual error. Reprinted with permission from SZ Li, SC Yan, HJ Zhang, QS Cheng. 2002. Multi-view face alignment using direct appearance models. *Proc 5th Int Conf Automatic Face Gesture Recogn*, pp. 309–314. Copyright ©2002, IEEE.

is orthogonal to $\mathbb{S}_t$ (projected back to $\mathbb{R}^L$) and $\dim(\mathbb{S}_t) \ll \dim(T)$, the dimension of space $\{\delta T\}$ is very high, and it is hard to train regression matrix $\mathbf{A}_a, \mathbf{A}_p$, and the prediction of the variance of position can be subject to significant errors. Also it is time and memory intensive. The TC-ASM is aimed at overcoming the above problems.

The TC-ASM consists of a shape model, a texture model, K local appearance models, and a *texture-constrained* shape model. The former three types are exactly the same as in the ASM and AAM. The texture-constrained shape model, or the mapping from texture to shape, is simply assumed linear and could be easily learned. In each step of the optimization, a better shape is found under a Bayesian framework. The details of the model will be introduced in the following.

5.1. Texture-Constrained Shape Model

In the shape model, there are some landmarks defined on the edges or contours. Since they have no explicit definition for their positions, there exists uncertainty of the shape given the texture, while there are correlations between the shape and the texture. The conditional distribution of shape parameters s given texture parameters t is simply assumed Gaussian, i.e.,

$$p(s|t) \sim N(s_t, \mathbf{\Sigma}_t), \tag{24}$$

where $\mathbf{\Sigma}_t$ stands for the covariance matrix of the distribution, and s_t is linearly determined by texture t. The linear mapping from t to s_t is:

$$s_t = \mathbf{R}t, \tag{25}$$

where $\mathbf{R}$ is a projection matrix that can be pre-computed from training pairs $\{(s_i, t_i)\}$ by singular-valued decomposition. For simplicity, $\mathbf{\Sigma}_t$ is assumed to be a known constant matrix. Figure 7 demonstrates the accuracy of the prediction in the test data via matrix $\mathbf{R}$. We may see that the predicted shape is close to the labeled shape even under varying illuminations. Thus, the constraints over the shape from the texture can be used as an evaluation criterion in the shape localization task. The prediction of matrix $\mathbf{R}$ is also affected by illumination variation, yet since Eq. (25) is formulated based on the eigenspace, the influence of the unfamiliar illumination can be alleviated when the texture is projected to the eigenspace.

Distribution Eq. (24) can also be represented as the prior distribution of s given shape s_t:

$$p(s|s_t) \propto \exp\{-Eng(s; s_t)\}, \tag{26}$$

where the energy function is:

$$Eng(s; s_t) = \|s - s_t\|^2_{\mathbf{\Sigma}_t}. \tag{27}$$

Figure 7. Comparison of the manually labeled shape (middle row) and the shape (bottom row) derived from the enclosed texture using the learned projection matrix: $s_t = \mathbf{R}t$. In the top row are the original images. All the images are test data. Reprinted with permission from SC Yan, C Liu, SZ Li, HJ Zhang, H Shum, QS Cheng. 2003. Face alignment using texture-constrained active shape models. *Image Vision Comput* **21**(1):69–75. Copyright ©2003, Elsevier.

5.2. TC-ASM in Bayesian Framework

The TC-ASM search begins with the mean shape, namely shape parameters $s^0 = 0$. The whole search process is outlined as below:

1. Set the iteration number $n = 1$;
2. Using the local appearance models in the ASM, we may obtain the candidate shape, S^n_{lm}, with shape parameters s^n_{lm} based on the shape, $S^{(n-1)}$, of the previous iteration;
3. The texture enclosed by S^n_{lm} is warped to the mean shape, denoted by t^n. The texture-constrained shape s^n_t is predicted from t^n by Eq. (25);
4. The *posterior* (MAP) estimation of S^n or s^n, given by S^n_{lm} and s^n_t, is derived based on the Bayesian framework;

5. If the stopping condition is satisfied, exit; otherwise, $n = n + 1$, goto step 2.

In the following we illustrate step 4 and the stopping condition in detail. To simplify the notation, we shall omit superscript n in the following deduction since the iteration number is constant. In step 4 the *posterior* (MAP) estimation of s, given by S_{lm} and s_t, is

$$p(s|S_{lm}, s_t) = \frac{p(S_{lm}|s, s_t)p(s, s_t)}{p(S_{lm}, s_t)}. \tag{28}$$

Assume that S_{lm} is conditionally independent from s_t, given s, i.e.,

$$p(S_{lm}|s, s_t) = p(S_{lm}|s). \tag{29}$$

Then

$$p(s|S_{lm}, s_t) \propto p(S_{lm}|s)p(s|s_t). \tag{30}$$

The corresponding energy function is

$$Eng(s; S_{lm}, s_t) = Eng(S_{lm}; s) + Eng(s; s_t). \tag{31}$$

From Eqs. (4) and (27), the best shape obtained in each step is

$$\begin{aligned} s &= \arg\min_s [Eng(s; S_{lm}) + Eng(s; s_t)] \\ &= \arg\min_s \|s_{lm} - s\|^2_{\mathbf{\Lambda}} + \|s - s_t\|^2_{\mathbf{\Sigma}_t} \\ &= \arg\min_s [s^T(\mathbf{\Lambda}^{-1} + \mathbf{\Sigma}_t^{-1})s - 2s^T(\mathbf{\Lambda}^{-1}s_{lm} + \mathbf{\Sigma}_t^{-1}s_t)] \\ &= (\mathbf{\Lambda}^{-1} + \mathbf{\Sigma}_t^{-1})^{-1}(\mathbf{\Lambda}^{-1}s_{lm} + \mathbf{\Sigma}_t^{-1}s_t). \end{aligned}$$

After restoring the superscript of iteration number, the best shape obtained in step n is

$$s^n = (\mathbf{\Lambda}^{-1} + \mathbf{\Sigma}_t^{-1})^{-1}(\mathbf{\Lambda}^{-1}s^n_{lm} + \mathbf{\Sigma}_t^{-1}s^n_t). \tag{32}$$

This indicates that the best shape derived in each step is an interpolation between the shape from the local appearance model and the texture-constrained shape. In this sense, the TC-ASM could be regarded as a tradeoff between the ASM and AAM methods.

The stopping condition of the optimization is: if the shape from the local appearance model and the texture-constrained shape are the same, i.e., the solution generated by ASM is verified by the AAM, the optimal solution must have been touched. In practice, however, these two shapes would hardly turn out to be the same. A threshold is introduced to evaluate the similarity, and sometimes the convergence criterion in the ASM is used (if the above criterion has not been satisfied for a long time). For higher efficiency and accuracy, a multi-resolution pyramid method is adopted in the optimization process.

6. EVALUATION FOR FACE ALIGNMENT

The emergence of many effective face alignment algorithms serves as a contrast to the lack of an effective method for evaluation of face alignment results. In the ASM, there has been no convergence criterion for the iteration. As such, the ASM search can give a bad result without giving the user a warning. In the AAM and DAM, the PCA reconstruction error is used as a distance measure for evaluation of alignment quality. However, the reconstruction error may not be a good discriminant for evaluation of the alignment quality because a non-face can look like a face when projected onto the PCA face subspace. In the TC-ASM, the algorithm claims to reach a convergence when the solution generated by the ASM is verified by the AAM, whereas both convergence criteria are not yet stable.

In this section we propose a statistical learning approach for constructing an evaluation function for face alignment. A *nonlinear* classification function is learned from a training set of positive and negative training examples to effectively distinguish between qualified and unqualified alignment results. The positive subset consists of qualified face alignment examples, and the negative subset consists of obviously unqualified and near-but-not-qualified examples.

We use the AdaBoost algorithm [25, 26] for learning. A set of candidate weak classifiers are created based on edge features extracted using Sobel- like operators. We choose to use edge features because crucial cues for alignment quality are around edges. Experimentally, we also found that the Sobel features produced significantly better results than other features, such as Haar wavelets. AdaBoost learning selects or learns a sequence of best features and the corresponding weak classifiers, and combines them into a strong classifier.

In the training stage, several strong classifiers are learned in stages using bootstrap training samples, and in the test they are cascaded to form a stronger classifier, following an idea in boosting-based face detection [27]. Such a divide-and-conquer strategy makes the training easier and the good–bad classification more effective. The evaluation function thus learned gives a quantitative confidence and the good–bad classification is achieved by comparing the confidence with a learned optimal threshold.

There are two important distinctions between evaluation functions thus learned and the linear evaluation function of reconstruction error used in the AAM. First, the evaluation is learned in such a way to distinguish between good and bad alignment. Second, the scoring is nonlinear, which provides a semantically more meaningful classification between good and bad alignment. Experimental results demonstrate that the classification function learned using the proposed approach provides semantically meaningful scoring for classification between qualified and unqualified face alignment.

6.1. Solution Quality Evaluation in ASM/AAM

There has been no convergence criterion for ASM search. In ASM search, the mean shape is placed near the center of the detected image and a coarse-to-fine search performed. Large movements are made in the first few iterations, getting the position roughly. As the search is progressing, more subtle adjustments are made. The result can yield a good match to the target image or it can fail (see Figure 8). Failure can happen even if the starting position is near the target. When the variations of expression and illumination are large, ASM search can diverge in order to match the local image pattern.

Figure 8. Four face instances of qualified (top) and unqualified (bottom) examples with their warped images. Reprinted with permission from XS Huang, SZ Li, YS Wang. 2004. Statistical learning of evaluation function for ASM/AAM image alignment. In *Proceedings: Biometric Authentication, ECCV 2004 International Workshop, BioAW 2004, Prague, Czech Republic, May 15, 2004* (ECCV Workshop BioAW), pp. 45– 56. Ed D Maltoni, AK Jain. New York: Springer. Copyright ©2004, Springer.

A similar problem exists in AAM search. There, the PCA reconstruction error is used as a distance measure for evaluation of alignment quality (and for guiding the search as well). However, the reconstruction error may not be a good discriminant for evaluation of alignment quality because a non-face can look like a face when projected onto the PCA face subspace. Cootes pointed out that, of 2700 testing examples, 519 failed to converge to a satisfactory result (the mean point position error is greater than 7.5 pixels per point) [4].

In the following we present a learning-based approach for the learning evaluation function for ASM/AAM based alignment.

6.2. AdaBoost-Based Learning

Our objective is to learn an evaluation function from a training set of qualified and unqualified alignment examples. From now on we use the terms "positive" and "negative" examples for classes of data. These examples are the face image after warping to a mean shape, as shown in Figure 8. Face alignment quality evaluation can be posed as a two-class classification problem: given an alignment result x (i.e., warped face), evaluation function $H(x) = +1$ if x is positive example, or -1 otherwise. We want to learn such an $H(x)$ that can provide a score in $[-1, +1]$ with a threshold around 0 for the binary classification.

For two-class problems, a set of N labeled training examples is given as $(x_1, y_1), \ldots, (x_N, y_N)$, where $y_i \in \{+1, -1\}$ is the class label associated with example $x_i \in \mathbb{R}^n$. A stronger classifier is a linear combination of M weak classifiers:

$$H_M(x) = \sum_{m=1}^{M} h_m(x). \tag{33}$$

In the real version of AdaBoost [25, 26], the weak classifiers can take a real value, $h_m(x) \in \mathbb{R}$, and have absorbed the coefficients needed in the discrete version ($h_m(x) \in -1, +1$ in the latter case). The class label for x is obtained as $H(x) = \text{sign}[H_M(x)]$, while magnitude $|H_M(x)|$ indicates the confidence. Every training example is associated with a weight. During the learning process, the weights are updated dynamically in such a way that more emphasis is placed on hard examples that are erroneously classified previously. It has been noted in recent studies [28, 29, 30] that the artificial operation of explicit re-weighting is unnecessary and can be incorporated into a functional optimization procedure of boosting.

An error occurs when $H(x) \neq y$, or $yH_M(x) < 0$. The "margin" of an example, (x, y), achieved by $h(x) \in \mathbb{R}$ on the training set examples is defined as $yh(x)$. This can be considered a measure of the confidence of h's prediction. The upper bound of classification error achieved by H_M can be derived as the following exponential loss function [31]:

$$J(H_M) = \sum_i \mathrm{e}^{-y_i H_M(x_i)} = \sum_i \mathrm{e}^{-y_i \sum_{m=1}^{M} h_m(x)}. \tag{34}$$

AdaBoost constructs $h_m(x)$ by stagewise minimization of Eq. (34). Given the current $H_{M-1}(x) = \sum_{m=1}^{M-1} h_m(x)$, the best $h_M(x)$ for the new strong classifier, $H_M(x) = H_{M-1}(x) + h_M(x)$ is the one that leads to the minimum cost:

$$h_M = \arg\min_{h^\dagger} J(H_{M-1}(x) + h^\dagger(x)). \tag{35}$$

0. (Input)
   (1) Training examples $\{(x_1, y_1), \ldots, (x_N, y_N)\}$,
   where $N = a + b$; of which a examples have $y_i = +1$
   and b examples have $y_i = -1$;
   (2) The maximum number $M_{\max}$ of weak classifiers to be combined;
1. (Initialization)
   $w_i^{(0)} = \frac{1}{2a}$ for those examples with $y_i = +1$ or
   $w_i^{(0)} = \frac{1}{2b}$ for those examples with $y_i = -1$.
   $M = 0$;
2. (Forward Inclusion)
   while $M < M_{\max}$
   (1) $M \leftarrow M + 1$;
   (2) Choose h_M according to Eq.36;
   (3) Update $w_i^{(M)} \leftarrow \exp[-y_i H_M(x_i)]$, and normalize to $\sum_i w_i^{(M)} = 1$;
3. (Output)
   $H(x) = \text{sign}[\sum_{m=1}^{M} h_m(x)]$.

Figure 9. AdaBoost algorithm. Reprinted with permission from XS Huang, SZ Li, YS Wang. 2004. Statistical learning of evaluation function for ASM/AAM image alignment. In *Proceedings: Biometric Authentication, ECCV 2004 International Workshop, BioAW 2004, Prague, Czech Republic, May 15, 2004* (ECCV Workshop BioAW), pp. 45–56. Ed D Maltoni, AK Jain. New York: Springer. Copyright ©2004, Springer.

The minimizer is [25, 26]

$$h_M(x) \quad = \quad \frac{1}{2} \log \frac{P(y = +1|x, w^{(M-1)})}{P(y = -1|x, w^{(M-1)})}, \tag{36}$$

where $w^{(M-1)}(x, y) = \exp(-yF_{M-1}(x))$ is the weight for the labeled example (x, y) and

$$P(y = +1|x, w^{(M-1)}) = \frac{E\left(w(x, y) \cdot 1_{[y=+1]}|x\right)}{E\left(w(x, y) \mid x\right)}, \tag{37}$$

where $E(\cdot)$ stands for the mathematical expectation and $1_{[C]}$ is 1 if C is true or 0 otherwise. $P(y = -1|x, w^{(M-1)})$ is defined similarly.

The AdaBoost algorithm based on the descriptions from [25, 26] is shown in Figure 9. There, the re-weight formula in step 2.3 is equivalent to the multiplicative rule in the original form of AdaBoost [32, 25]. In Section 6.3, we will present a statistical model for stagewise approximation of $P(y = +1|x, w^{(M-1)})$.

6.3. Construction of Candidate Weak Classifiers

The optimal weak classifier at stage M is derived as Eq. (36). Using $P(y|x,w)= p(x|y,w)P(y)$, it can be expressed as

$$h_M(x) = L_M(x) - T, \tag{38}$$

where

$$L_M(x) = \frac{1}{2}\log\frac{p(x|y=+1,w)}{p(x|y=-1,w)}, \tag{39}$$

$$T = \frac{1}{2}\log\frac{P(y=+1)}{P(y=-1)}. \tag{40}$$

The log likelihood ratio (LLR), $L_M(x)$, is learned from the training examples of the two classes. The threshold T is determined by the log ratio of prior probabilities. In practice, T can be adjusted to balance between the detection and false alarm rates (i.e., to choose a point on the ROC curve).

Learning optimal weak classifiers requires modeling the LLR of Eq. (39). Estimating the likelihood for high-dimensional data x is a non-trivial task. In this work, we make use of the stagewise characteristics of boosting, and derive likelihood $p(x|y,w^{(M-1)})$ based on an over-complete scalar feature set $\mathcal{Z} = \{z'_1,\ldots,z'_K\}$. More specifically, we approximate $p(x|y,w^{(M-1)})$ by $p(z_1, \ldots, z_{M-1}, z'|y, w^{(M-1)})$, where z_m $(m = 1,\ldots,M-1)$ are the features that have already been selected from $\mathcal{Z}$ by the previous stages, and z' is the feature to be selected. The following describes the candidate feature set $\mathcal{Z}$, and presents a method for constructing weak classifiers based on these features.

Because the shape is about boundaries between regions, it makes sense to use edge information (magnitude or orientation or both) extracted from a grayscale image. In this work, we use a simple Sobel filter for extracting the edge information. Two filters are used: K_w for horizontal edges and K_h for vertical edges, as follows:

$$K_w(w,h) = \begin{pmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{pmatrix} \quad and \quad K_h(w,h) = \begin{pmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{pmatrix}. \tag{41}$$

The convolution of the image with the two filter masks gives two edge strength values:

$$G_w(w,h) = K_w * I(w,h), \tag{42}$$

$$G_h(w,h) = K_h * I(w,h), \tag{43}$$

The edge magnitude and direction are obtained as

$$S(w,h) = \sqrt{G_w^2(w,h) + G_h^2(w,h)}, \tag{44}$$

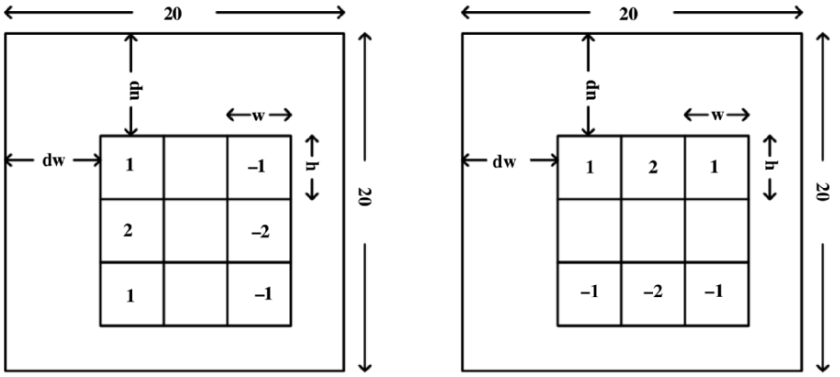

Figure 10. The two types of simple Sobel-like filters defined on sub-windows. The rectangles are of size $w \times h$ and are at distances of (dw, dh) apart. Each feature takes a value calculated by the weighted $(\pm 1, \pm 2)$ sum of the pixels in the rectangles. Reprinted with permission from XS Huang, SZ Li, YS Wang. 2004. Statistical learning of evaluation function for ASM/AAM image alignment. In *Proceedings: Biometric Authentication, ECCV 2004 International Workshop, BioAW 2004, Prague, Czech Republic, May 15, 2004* (ECCV Workshop BioAW), pp. 45–56. Ed D Maltoni, AK Jain. New York: Springer. Copyright ©2004, Springer.

$$\phi(w, h) = \arctan\left(\frac{G_h(w, h)}{G_w(w, h)}\right). \tag{45}$$

The edge information based on the Sobel operator is sensitive to noise. To solve this problem we use the sub-block of the image to convolve with the Sobel filter (see Figure 10), which is similar to Haar-like feature calculation.

6.4. Statistical Learning of Weak Classifiers

A scalar feature $z'_k : x \to \mathbf{R}$ is a transform from the n-dimensional (400D if a face example x is of size 20 $\times$ 20) data space to the real line. These block differences are an extension to the Sobel filters. For each face example of size 20 $\times$ 20, there are hundreds of thousands of different z'_k for admissible w, h, dw, dh values, so $\mathcal{Z}$ is an over-complete feature set for the intrinsically low-dimensional face pattern x. In this work, an optimal weak classifier (38) is associated with a single scalar feature; to find the best new weak classifier is to choose the best corresponding feature.

We can define the following component LLRs for target $L_M(x)$:

$$\tilde{L}_m(x) \quad = \quad \frac{1}{2} \log \frac{p(z_m | y = +1, w^{(m-1)})}{p(z_m | y = -1, w^{(m-1)})} \tag{46}$$

for the selected features, z_m's ($m = 1, \ldots, M-1$), and

$$L_k^{(M)}(x) = \frac{1}{2} \log \frac{p(z'_k(x)|y=+1, w^{(M-1)})}{p(z'_k(x)|y=-1, w^{(M-1)})} \tag{47}$$

for features to be selected, $z'_k \in \mathcal{Z}$. Then, after some mathematical derivation, we can approximate the target LLR function as

$$L_M(x) = \frac{1}{2} \log \frac{p(x|y=+1, w^{(M-1)})}{p(x|y=-1, w^{(M-1)})} \approx \sum_{m=1}^{M-1} \tilde{L}_m(x) + L_k^{(M)}(x). \tag{48}$$

Let

$$\Delta L_M(x) = L_M(x) - \sum_{m=1}^{M-1} \tilde{L}_m(x). \tag{49}$$

The best feature is the one whose corresponding $L_k^{(M)}(x)$ best fits $\Delta L_M(x)$. It can be found as the solution to the following minimization problem:

$$k^* = \arg\min_{k,\beta} \sum_{i=1}^{N} \left[\Delta L_M(x_i) - \beta L_k^{(M)}(x_i)\right]^2. \tag{50}$$

This can be done in two steps as follows. First, find k^* for which

$$(L_k^{(M)}(x_1), L_k^{(M)}(x_2), \ldots, L_k^{(M)}(x_N)) \tag{51}$$

is most parallel to

$$(\Delta L_M(x_1), \Delta L_M(x_2), \ldots, \Delta L_M(x_N)). \tag{52}$$

This amounts to finding k for which $L_k^{(M)}$ is most correlated with ΔL_M over the data distribution, and set $z_M = z'_{k^*}$. Then, we compute

$$\beta^* = \frac{\sum_{i=1}^{N} \Delta L_M(x_i) L_{k^*}(x_i)}{\sum_{i=1}^{N} [L_{k^*}(x_i)]^2}. \tag{53}$$

After that, we obtain

$$\tilde{L}_M(x) = \beta^* L_{k^*}(x). \tag{54}$$

The strong classifier is then given as

$$H_M(x) = \sum_{m=1}^{M} \left(\tilde{L}_m(x) - T\right) \tag{55}$$

$$= \sum_{m=1}^{M} \tilde{L}_m(x) - MT. \tag{56}$$

The evaluation function $H_M(x)$ thus learned gives a quantitative confidence and the good–bad classification is achieved by comparing the confidence with the threshold value of zero.

7. EXPERIMENTAL RESULTS

7.1. DAM

7.1.1. Computation of Subspaces

A total of 80 images of size 128×128 are collected. Each image contains a different face in an area of about 64×64 pixels. The images set are randomly partitioned into a training set of 40 images and a test set of the other 40. Each image is mirrored, and this doubles the total number of images in each set.

$K = 72$ face landmark points are labeled manually (see an example in Figure 11. The shape subspace is $k = 39$ dimensional, which retains 98% of the total shape variation. The mean shape contains a texture of $L = 3186$ pixels. The texture subspace is $\ell = 72$ dimensional, as the result of retaining 98% of total texture variation. These are common to both the AAM and DAM.

For the AAM, an appearance subspace is constructed to combine both shape and texture information. A concatenated shape and texture vector is 39 + 72 dimensional, where the weight parameter is calculated as $r = 7.5$ for $\mathbf{\Lambda} = r\mathbf{I}$ in Eq. (7). It is reduced to a 65-dimensional appearance subspace that retains 98% of total variation of the concatenated features.

For the DAM, the linearity assumption made for the model, $s = \mathbf{R}t + \varepsilon$, of Eq. (16) is well verified because all the elements in $E(\varepsilon\varepsilon^T)$ calculated over the training set are smaller than 10^{-5}.

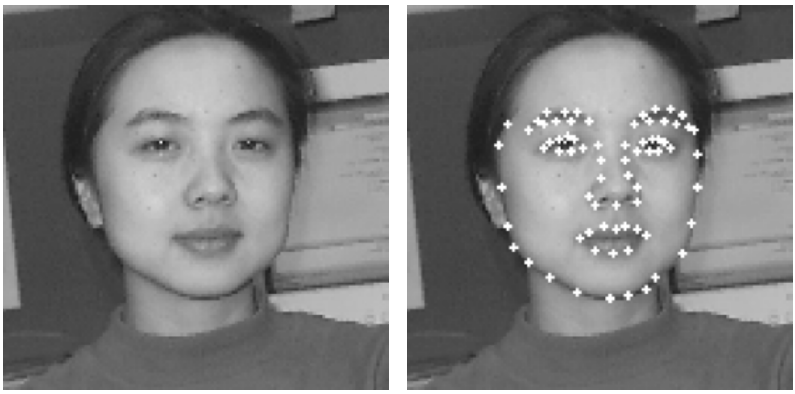

Figure 11. A face image and the landmark points. Reprinted with permission from XW Hou, SZ Li, HJ Zhang, QS Cheng. 2001. Direct appearance models. *Proc IEEE Conf Comput Vision Pattern Recogn* **1**:828–833. Copyright ©2001, IEEE.

The original texture difference δT, which is used in the AAM for predicating position displacement, is 3186 dimensional; it is reduced to 724-dimensional $\delta T'$, which is used in the DAM for prediction, to retain 98% of variation over the 1920 training examples.

The DAM requires much less memory during learning of prediction matrices $\mathbf{R}_p$ in Eq. (22) than AAM for learning $\mathbf{A}_a$ in Eq. (11). For the DAM, there are 80 training images, 4 parameters for the position $(x, y, \theta, scale)$, and 6 disturbances for each parameter to generate training data for training $\mathbf{R}_p$. So, the size of training data for the DAM is $80 \times 4 \times 6 = 1920$. For the AAM, there are 80 training images, 65 appearance parameters, and 4 disturbances for each parameter to generate training data for training $\mathbf{A}_a$. The size of the training data set for $\mathbf{A}_a$ is $80 \times 65 \times 4 = 20800$. Therefore, the size of the training data set for AAM's prediction matrices is $20800 + 1920 = 22720$, which is 11.83 times that for the DAM. On a PC, for example, the memory capacity for AAM training with 80 images would allow DAM training with 946 images.

7.1.2. Alignment and Appearance Estimation

Table 2 compares the DAM and AAM in terms of the quality of position and texture parameter estimates, and the convergence rates. The effect of using $\delta T'$ instead of δT is demonstrated through DAM$'$, which is DAM minus the PCA subspace modeling of δT. The initial position is a shift from the true position by $dx = 6, dy = 6$. The $\|\delta p\|$ is calculated for each image as the averaged distance between corresponding points in the two shapes, and therefore it is also a measure

Table 2. Comparisons of DAM, DAM' and AAM in terms of errors in estimated texture (appearance) parameters δT and position δp and convergence rates for the training images (first block of three rows) and test images (second block)

	$E(\|\delta T\|^2)$	std$(\|\delta T\|^2)$	$E(\|\delta p\|)$	std$(\|\delta p\|)$	cvg rate
DAM	0.156572	0.065024	0.986815	0.283375	100%
DAM'	0.155651	0.058994	0.963054	0.292493	100%
AAM	0.712095	0.642727	2.095902	1.221458	70%
DAM	1.114020	4.748753	2.942606	2.023033	85%
DAM'	1.180690	5.062784	3.034340	2.398411	80%
AAM	2.508195	5.841266	4.253023	5.118888	62%

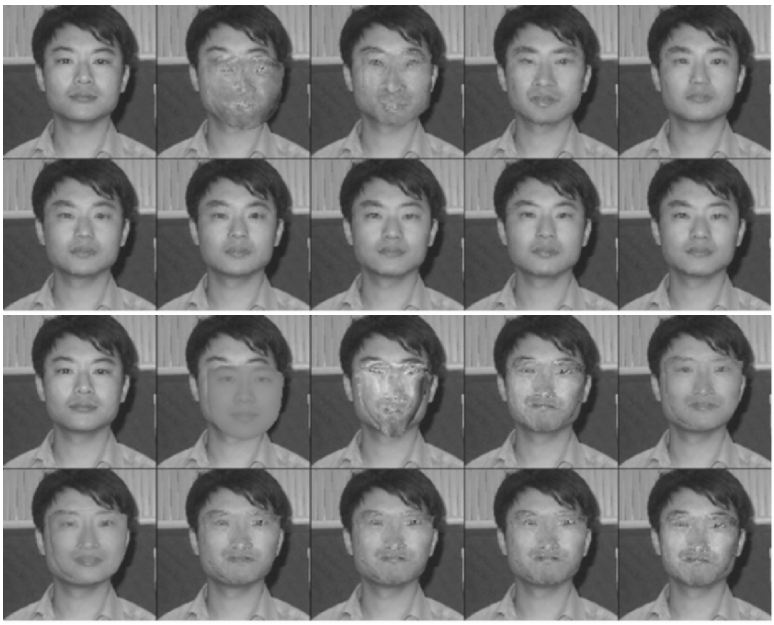

Figure 12. Scenarios of DAM (top) and AAM (bottom) alignment. Reprinted with permission from XW Hou, SZ Li, HJ Zhang, QS Cheng. 2001. Direct appearance models. *Proc IEEE Conf Comput Vision Pattern Recogn* **1**:828–833. Copyright ©2001, IEEE.

of difference in shape. The convergence is judged by satisfaction of two conditions: $\|\delta T\|^2 < 0.5$ and $\|\delta p\| < 3$.

Figure 12 illustrates average scenarios of DAM and AAM alignment. Figure 13 illustrates the dynamics of total error δT for 10 images randomly selected from the training set and 10 from the test set. We see that the DAM has faster convergence and smaller error than the AAM.

7.1.3. Multi-View DAM

The training set contains 200 frontal, 200 half-side, and 170 full-side view faces whose sizes are of about 64×64 pixels, while the test set contains 80 images for each view group. The landmark points are labeled manually (see Figure 2 and Table 1). They are used for the training and as ground-truth in the test stage.

To compare, we also implemented the AAM using the same data in the frontal view. The shape and texture parameter vectors are 69 + 144 dimensional, respectively, where the weight parameter for the concatenation of the two parts is calculated as $r = 8.84$ for $\mathbf{\Lambda} = r\mathbf{I}$ in Eq. (7). The concatenated vector space is reduced to a 113-dimensional appearance subspace that retains 98% of the total variation of the concatenated features.

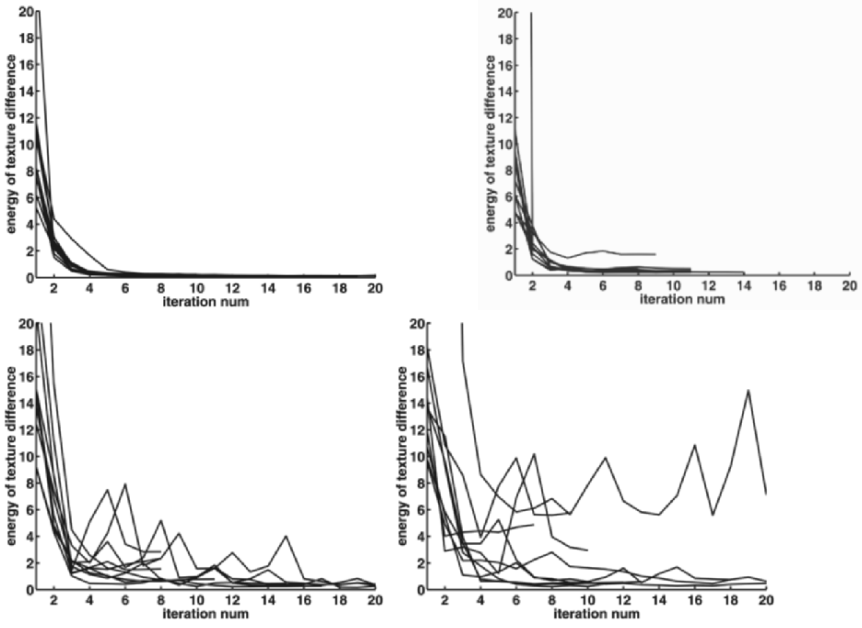

Figure 13. The evolution of total δT for the DAM (top) and AAM (bottom) as a function of iteration number for the training (left) and test (right) images. Reprinted with permission from XW Hou, SZ Li, HJ Zhang, QS Cheng. 2001. Direct appearance models. *Proc IEEE Conf Comput Vision Pattern Recogn* **1**:828–833. Copyright ©2001, IEEE.

Some results about DAM learning and search have been presented in Figure 2–6. Figure 14 compares the convergence rate and accuracy properties of the DAM and AAM (for the frontal view) in terms of the error in δT (cf. Eq. (10)) as the algorithms iterate. The statistics are calculated from 80 images randomly selected from the training set and 80 images from the test set. We can see that the DAM has faster a convergence rate and smaller error than the AAM. Figure 15 illustrates the error of DAM for non-frontal faces. Figure 16 compares the alignment accuracy of the DAM and AAM (for frontal faces) in terms of the percentage of images whose texture reconstruction error δT is smaller than 0.2, where the statistics are obtained using another test set including the 80 test images mentioned above and an additional 20 other test images. It shows again that the DAM is more accurate than the AAM.

The DAM search is fairly fast. It takes on average 39 ms per iteration for frontal and half-side view faces, and 24 ms for full-side view faces in an image of size 320×240 pixels. Every view model takes about 10 iterations to converge. If 3 view models are searched per face, as is done with image sequences from video, the algorithm takes about 1 second to find the best face alignment.

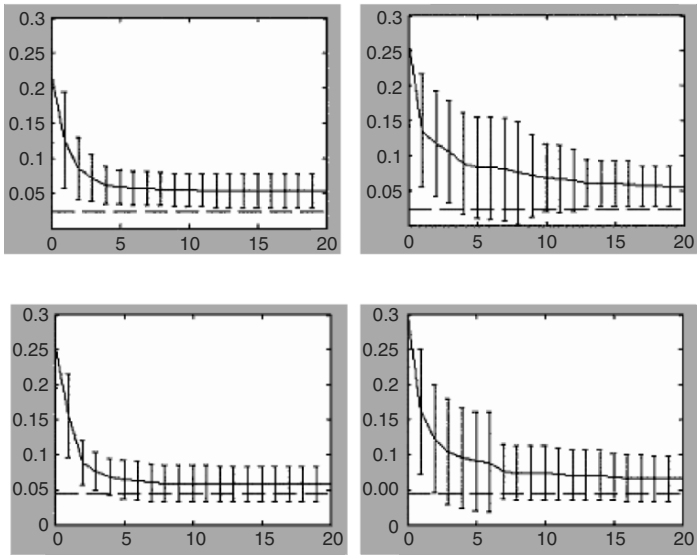

Figure 14. Mean error (the curve) and standard deviation (the bars) in reconstructed texture $\|\delta T\|$ as a function of iteration number for DAM (left) and AAM (right) methods with the training (top) and test (bottom) sets, for frontal face images. The horizontal dashed lines in the lower part of the figures indicate average $\|\delta T\|$ for the manually labeled alignment. Reprinted with permission from SZ Li, SC Yan, HJ Zhang, QS Cheng. 2002. Multi-view face alignment using direct appearance models. *Proc 5th Int Conf Automatic Face Gesture Recogn*, pp. 309– 314. Copyright ©2002, IEEE.

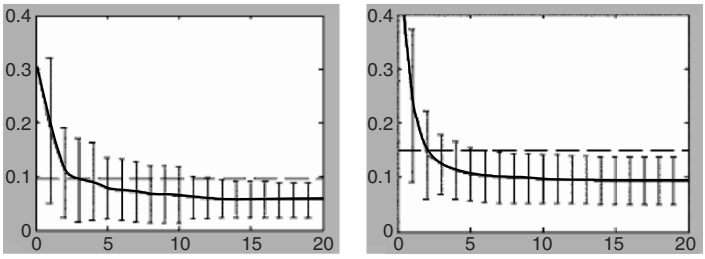

Figure 15. Mean error in $\|\delta T\|$ and standard deviation of DAM alignment for half- (left) and full- (right) side view face images from the test set. Note that the mean errors in the calculated solutions are smaller than obtained using the manually labeled alignment after a few iterations. Reprinted with permission from SZ Li, SC Yan, HJ Zhang, QS Cheng. 2002. Multi-view face alignment using direct appearance models. *Proc 5th Int Conf Automatic Face Gesture Recogn*, pp. 309–314. Copyright ©2002, IEEE.

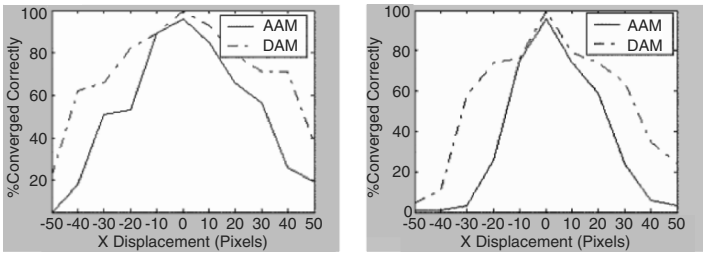

Figure 16. Alignment accuracy of the DAM (dashed) and AAM (solid) in terms of localization errors in the x (left) and y (right) directions. Reprinted with permission from SZ Li, SC Yan, HJ Zhang, QS Cheng. 2002. Multi-view face alignment using direct appearance models. *Proc 5th Int Conf Automatic Face Gesture Recogn*, pp. 309–314. Copyright ©2002, IEEE.

7.2. TC-ASM

A data set containing 700 face images with different illumination conditions and expressions are selected from the AR database [33] in our experiments, each of which is 512×512, 256 gray images containing the frontal view face about 200×200. 83 landmark points are manually labeled on the face. We randomly select 600 for training and the other 100 for testing.

For comparison, the ASM and AAM are trained on the same data sets, in a three-level image pyramid (resolution is reduced 1/2 level by level) as with the TC-ASM. By means of PCA with 98% total variations retained, the dimension of the shape parameter in the ASM shape space is reduced to 88, and the texture parameter vector in the AAM texture space is reduced to 393. The concatenated vector of the shape and texture parameter vectors with the weighting parameter, $\gamma = 13.77$, is reduced to 277. Two types of experiments are presented: (1) comparison of the point-position accuracy, and (2) comparison of the texture reconstruction error. The experiments are all performed in the 3-level resolution image pyramid.

7.2.1. Point Position Accuracy

The average point-point distances between the searched shape and the manually labeled shape of the three models are compared in Figure 17. The vertical axis represents the percentage of the solutions for which the average point-to-point distances to the manually labeled ones are smaller than the corresponding horizonal axis value. The statistics are calculated from 100 test images with different initializations, with random displacements to the ground truth of 10, 20, 30, and 40 pixels. The results show that TC-ASM outperforms both the ASM and AAM in most cases since the TC-ASM curve lies above the ASM and AAM curves. It also suggests that the AAM outperforms the ASM when the initial displacement is small, while the ASM is more robust with an increasing initial displacement.

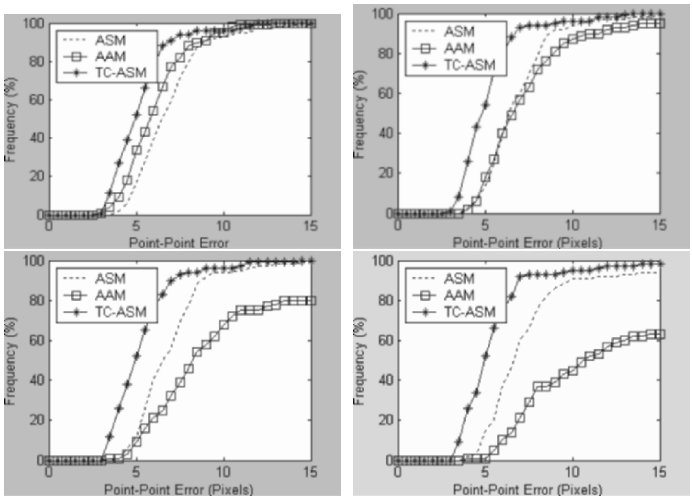

Figure 17. Accuracy of ASM, AAM, TC-ASM. From upper to lower, left to right, are the results obtained with the initial displacements of 10, 20, 30, and 40 pixels. Note that the value of the vertical coordinate is the percentage of examples that have the point-to-point distance smaller than the corresponding value of horizonal coordinate. Reprinted with permission from SC Yan, C Liu, SZ Li, HJ Zhang, H Shum, QS Cheng. 2003. Face alignment using texture-constrained active shape models. *Image Vision Comput* **21**(1):69–75. Copyright ©2003, Elsevier.

We compare the stability of the TC-ASM with the ASM in Figure 18. The value on the horizonal axis is the index number of the selected examples, whereas the value on the vertical axis is the average standard deviation of the results obtained from 10 different initializations that deviate from the ground-truth by approximately 20 pixels. The results are convincing that the TC-ASM is more stable to initialization. An example is given in Figure 19.

7.2.2. Texture Reconstruction Error

The texture reconstruction error comparison of the three models in Figure 20 illustrates that the TC-ASM improves the accuracy of texture matching. The texture accuracy of the TC-ASM is close to that of the AAM, while its position accuracy is better than that of the AAM (see Figure 17). Although the AAM has more cases with small texture reconstruction errors, the TC-ASM has more cases with a texture reconstruction error smaller than 0.2.

An example in which the AAM fails for a different illumination condition from the training data, yet the TC-ASM performs well, is presented in Figure 21. Figure 22 shows a scenario of AAM and TC-ASM alignment.

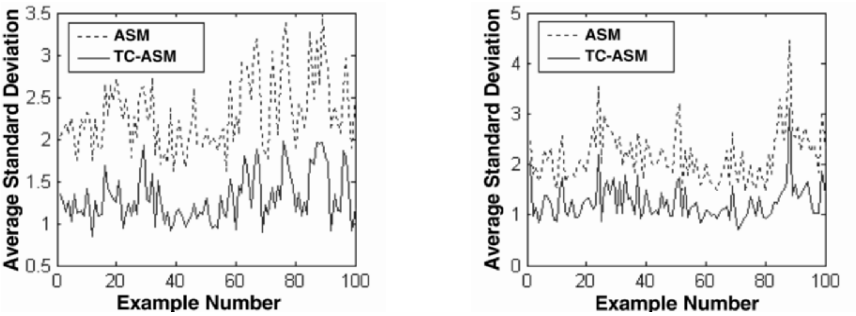

Figure 18. Standard deviation in the results of each example for ASM (dotted) and TC-ASM (solid) with the training set (left) and the test set (right). Reprinted with permission from SC Yan, C Liu, SZ Li, HJ Zhang, H Shum, QS Cheng. 2003. Face alignment using texture-constrained active shape models. *Image Vision Comput* **21**(1):69–75. Copyright ©2003, Elsevier.

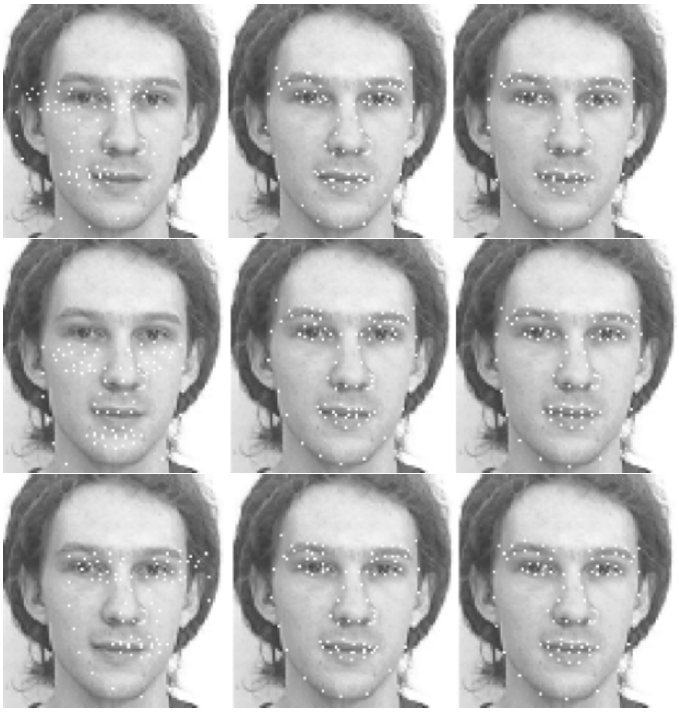

Figure 19. Stability of the ASM (middle column) and the TC-ASM (right column) in shape localization. The different initialization conditions are shown in the left column. Reprinted with permission from SC Yan, C Liu, SZ Li, HJ Zhang, H Shum, QS Cheng. 2003. Face alignment using texture-constrained active shape models. *Image Vision Comput* **21**(1):69–75. Copyright ©2003, Elsevier.

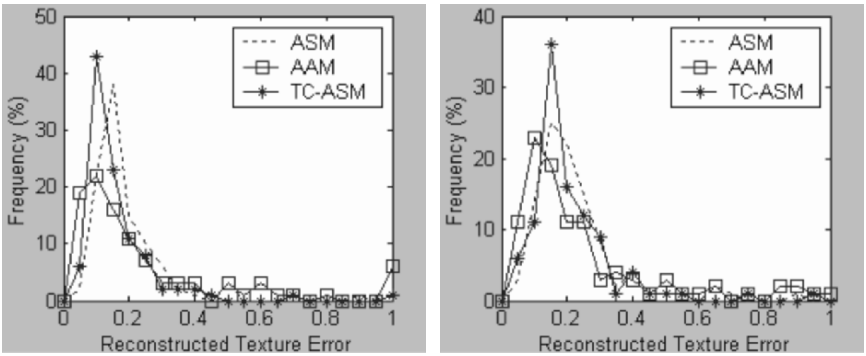

Figure 20. Distribution of the texture reconstruction error with the ASM (dotted), the AAM (square), and the TC-ASM (asterisk), with training data (left) and test data (right). Reprinted with permission from SC Yan, C Liu, SZ Li, HJ Zhang, H Shum, QS Cheng. 2003. Face alignment using texture-constrained active shape models. *Image Vision Comput* **21**(1):69–75. Copyright ©2003, Elsevier.

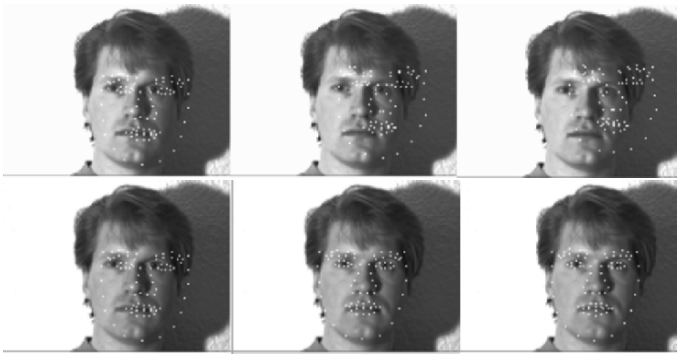

Figure 21. Sensitivities of the AAM (upper) and TC-ASM (lower) to an illumination condition not seen in the training data. From left to right are the results obtained at the 0th, 2th, and 10th iterations. Result in different levels of image pyramid is scaled back to the original scale. Reprinted with permission from SC Yan, C Liu, SZ Li, HJ Zhang, H Shum, QS Cheng. 2003. Face alignment using texture-constrained active shape models. *Image Vision Comput* **21**(1):69–75. Copyright ©2003, Elsevier.

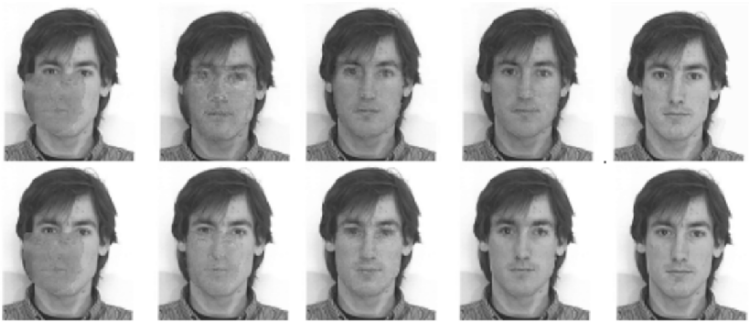

Figure 22. Scenarios of AAM (upper) and TC-ASM (lower) alignment with texture reconstruct errors 0.3405 and 0.1827, respectively. From left to right are the results obtained at the 0th, 5th, 10th, and 15th iterations, and the original image. Result in different levels of image pyramid is scaled back to the original scale. Reprinted with permission from SC Yan, C Liu, SZ Li, HJ Zhang, H Shum, QS Cheng. 2003. Face alignment using texture-constrained active shape models. *Image Vision Comput* **21**(1):69–75. Copyright ©2003, Elsevier.

From the experiment, the TC-ASM is more computationally expensive than ASM, but it is much faster than the AAM. In our experiment (600 training images, 83 landmarks, using a P-III 667 computer with 256Mb memory), it takes an average of 32 ms per iteration, which is twice that of the ASM (16 ms) but a fifth of the AAM (172 ms). The training time with the AAM is more than 2 hr, while for the TC-ASM it is only about 12 minutes.

7.3. Evaluation for Face Alignment

The positive and negative training and set data are generated as follows. All the shapes are aligned or are warping to the tangent space of the mean shape, $\overline{S}$. After that, the texture T_0 is warped correspondingly to $T \in \mathbb{R}^L$, where L is the number of pixels in the mean shape $\overline{S}$.

In our work, 2536 positive examples and 3000 negative examples are used to train a strong classifier. The 2536 positive examples are derived from 1268 original positive examples plus the mirror images. The negative examples are generated by random rotating, scaling, and shifting positive example shape points. A strong classifier is trained to reject 92% of the negative examples while correctly accepting 100% of the positive examples.

A cascade of classifiers is trained to train a computationally effective model, and makes training easier with a divide-and-conquer strategy. When training a new stage, negative examples are bootstrapped based on the classifier trained in the previous stages. The details of training a cascade of 5 stages is summarized

Table 3. Training results (WC: weak classifier)

stage	number of pos	number of neg	number of WC	False Alarm
1	2536	3000	22	0.076
2	2536	3000	237	0.069
3	2536	888	294	0.263
4	2536	235	263	0.409
5	2536	96	208	0.0

in Table 3. As the result of training, we achieved 100% correct acceptance and correct rejection rates on the training set.

We compare the learned evaluation function with the PCA texture reconstruction error-based evaluation method, using the same data sets (but PCA does not require negative examples in training). The dimensionality of the PCA subspace is chosen to retain 99% of the total variance of the data. The best threshold of reconstruction error is selected to minimize the classification error. Figure 23 shows the ROC curve for the reconstruction error-based alignment evaluation method for the training set. Note that this method cannot achieve 100% correct rates.

During the test, a total of 1528 aligned examples (800 qualified and 728 non-qualified images) are used. We evaluate each face images and give a score in terms of (a) the confidence value $H_M(x)$ for the learning-based method and (b) the confidence value $\mathrm{dist}_{PCA} - \mathrm{threshold}$ for the PCA-based method. The qualified and unqualified alignment decisions are judged by comparing the score with the normalized threshold of 0. Some examples of accepted (top part) and rejected (bottom part) face alignment results are shown in Figure 24. Figure 25 compares the two methods in terms of their ROC curves (first plot) and error curves (the second plot), where the axis label P (pos/neg) represents the false positive rate and so on.

Finally, we would like to mention that, experimentally, we also found that the Sobel features produced significantly better results than other features such as Haar wavelets. This is not elaborated here.

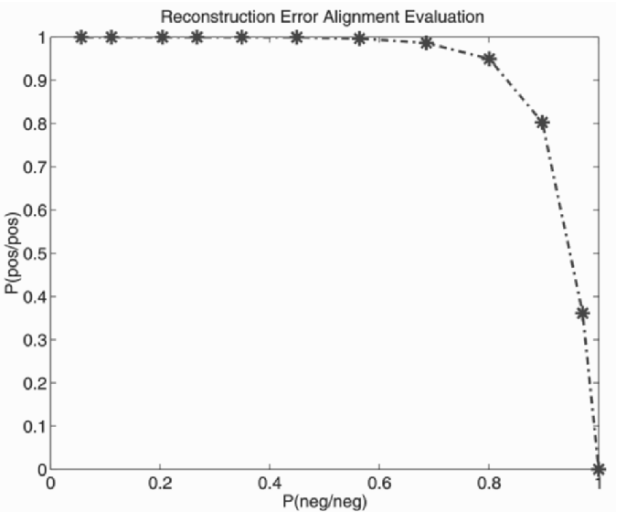

Figure 23. ROC curve for the reconstruction error-based alignment evaluation for the training set. Reprinted with permission from XS Huang, SZ Li, YS Wang. 2004. Statistical learning of evaluation function for ASM/AAM image alignment. In *Proceedings: Biometric Authentication, ECCV 2004 International Workshop, BioAW 2004, Prague, Czech Republic, May 15, 2004* (ECCV Workshop BioAW), pp. 45–56. Ed D Maltoni, AK Jain. New York: Springer. Copyright ©2004, Springer. See attached CD for color version.

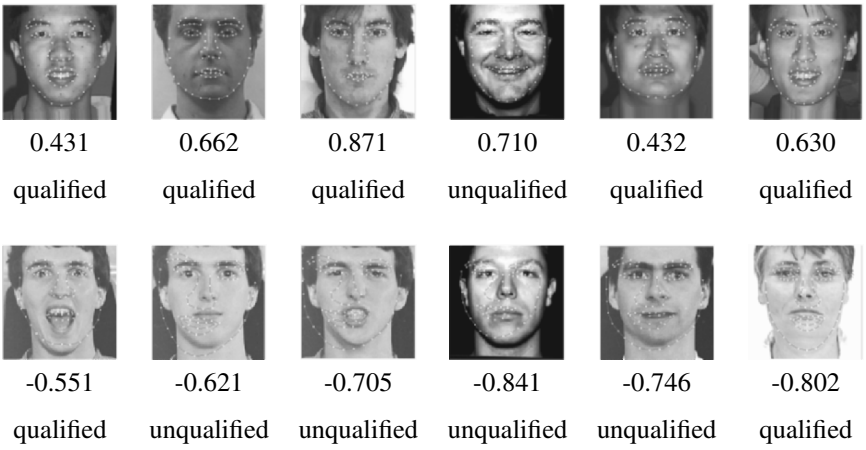

Figure 24. Alignment quality evaluation results: accepted images (top) and rejected images (bottom). Reprinted with permission from XS Huang, SZ Li, YS Wang. 2004. Statistical learning of evaluation function for ASM/AAM image alignment. In *Proceedings: Biometric Authentication*, ECCV 2004 International Workshop, BioAW 2004, Prague, Czech Republic, May 15, 2004 (ECCV Workshop BioAW), pp. 45–56. Ed D Maltoni, AK Jain. New York: Springer. Copyright ©2004, Springer. See attached CD for color version.

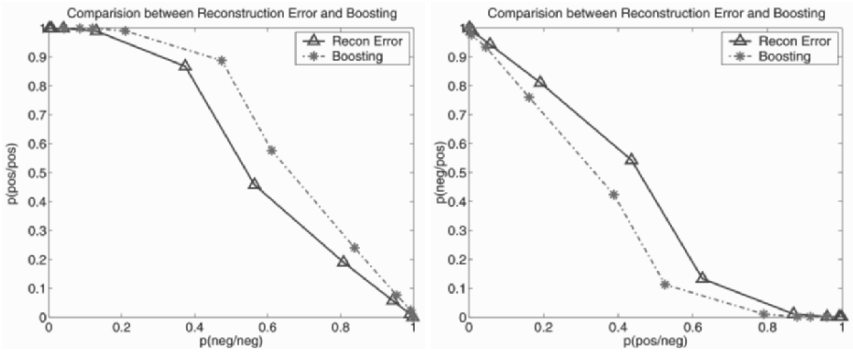

Figure 25. Comparison between reconstruction error method and boost method. Reprinted with permission from XS Huang, SZ Li, YS Wang. 2004. Statistical learning of evaluation function for ASM/AAM image alignment. In *Proceedings: Biometric Authentication, ECCV 2004 International Workshop, BioAW 2004, Prague, Czech Republic, May 15, 2004* (ECCV Workshop BioAW), pp. 45–56. Ed D Maltoni, AK Jain. New York: Springer. Copyright ©2004, Springer. See attached CD for color version.

8. CONCLUSION

In this chapter we reviewed important shape- and texture-based deformable models — such as the ASM, the AAM, and their variants — for image analysis. These image analysis tools not only provide alignment between the input and the target to best fit the constraints, but also provide aligned features for object pattern classification.

Although great advances have been made in the past decade, there remain challenges for future research. One area is the robustness of deformable models toward variance of pose, illumination, and expression. Existing models can only deal with a moderate amount of such variations, so performance deteriorates when extreme illumination conditions or exaggerated expressions are present. While a morphable model has demonstrated its effectiveness with 3D object analysis, efficient, real-time, and exact model searching algorithms are still lacking. Solving these problems will lead to better applications.

9. ACKNOWLEDGMENTS

This work was supported by the following funding: National Science Foundation of China Project #60518002, Chinese National 863 Program Projects #2004AA1Z2290 and #2004AA119050.

10. NOTES

1. It is a deviation of the mostly used energy function with a squared Euclidean distance between S_{lm}^n and shape $S \in \mathbb{R}^{2K}$ derived from parameter s. It is more reasonable to take into account the prior distribution in the shape space.

11. REFERENCES

1. Cootes TF, Taylor CJ, Cooper DH, Graham J. 1995. Active shape models: their training and application. *Comput Vision Image Understand* **61**:38–59.
2. Cootes TF, Edwards GJ, Taylor CJ. 1998. Active appearance models. In *Proceedings of the European conference on computer vision*, Vol. 2, pp. 484–498. Ed. H Burkhardt, B Neumann. New York: Springer.
3. Edwards GJ, Cootes TF, Taylor CJ. 1998. Face recognition using active appearance models. In *Proceedings of the European conference on computer vision*, Vol. 2, pp. 581–695. Ed. H Burkhardt, B Neumann. New York: Springer.
4. Cootes TF, Taylor CJ. 2001. *Statistical models of appearance for computer vision*. Technical Report, Wolfson Image Analysis Unit, Manchester University, www.isbe.man.ac.uk/simbim/refs.html.
5. Sclaroff S, Isidoro J. 1998. Active blobs. In *Proc IEEE Int Conf Comput Vision, Bombay, India*, pp. 1146–1153.
6. Cootes TF, Walker KN, Taylor CJ. 2000. View-based active appearance models. In *4th International conference on automatic face and gesture recognition, Grenoble, France*, pp. 227–232. http://citeseer.ist.psu.edu/cootes00viewbased.html.
7. Psarrou A, Romdhani S, Gong S. 1999. Learning a single active face shape model across views. In *Proceedings of the IEEE international workshop on recognition, analysis, and tracking of faces and gestures in real-time systems, Corfu, Greece, 26–27 September*, pp. 31–38. Washington, DC: IEEE Computer Society.
8. Cootes TF, Taylor CJ. 2001. Constrained active appearance models. *Proc IEEE Int Conf Comput Vision* **1**:748–754.
9. Blanz V, Vetter T. 1999. A morphable model for the synthesis of 3D faces. In *Proc. Siggraph'99*, pp. 187–194. New York: ACM Press.
10. Li Y, Gong S, Liddell and H. 2001. Constructing facial identity surfaces in a nonlinear discriminating space. *Proc IEEE Comput Soc Conf: Computer Vision and Pattern Recognition* **2**:258–263.
11. Duta N, Jain AK, Dubuisson-Jolly M. 2001. Automatic construction of 2D shape models. *IEEE Trans Pattern Analy Machine Intell* **23**(5):433–446.
12. van Ginneken B, Frangi AF, Staal JJ, ter Haar Romeny BM, Viergever MA. 2001. A nonlinear gray-level appearance model improves active shape model segmentation. In *IEEE workshop on mathematical models in biomedical image analysis*, pp. 205–212. Ed. L Staib, A Rangarajan. Washington, DC: IEEE Society Press.
13. Baker S, Matthews I. 2001. Equivalence and efficiency of image alignment algorithms. *Proc IEEE Conf Comput Vision Pattern Recognition* **1**:1090–1097.
14. Ahlberg J. 2001. Using the active appearance algorithm for face and facial feature tracking. In *IEEE ICCV workshop on recognition, analysis and tracking of faces and gestures in real-time systems, Vancouver, Canada, July 13, 2001*, pp. 68–72. Washington, DC: IEEE.
15. Hou XW, Li SZ, Zhang HJ, Cheng QS. 2001. Direct appearance models. *Proc IEEE Conf Comput Vision Pattern Recognition* **1**:828– 833.
16. Li SZ, Yan SC, Zhang HJ, Cheng QS. 2002. Multi-view face alignment using direct appearance models. *Proc 5th Int Conf Automatic Face Gesture Recogn, Washington, DC, 20–21 May 2002*, pp. 309–314. Washington, DC: IEEE.

17. Yan SC, Liu C, Li SZ, Zhang HJ, Shum H, Cheng QS. 2003. Face alignment using texture-constrained active shape models. *Image Vision Comput* **21**(1):69–75.
18. XS Huang, SZ Li, YS Wang. 2004. Statistical learning of evaluation function for ASM/AAM image alignment. In *Proceedings: Biometric Authentication, ECCV 2004 International Workshop, BioAW 2004, Prague, Czech Republic, May 15, 2004* (ECCV Workshop BioAW), pp. 45–56. Ed D Maltoni, AK Jain. New York: Springer.
19. Cootes TF, Edwards GJ, Taylor CJ. 1999. Comparing active shape models with active appearance models. In *10th British Machine Vison Conference*, Vol. 1, pp. 173–182. Ed. T Pridmore, D Elliman. Nottingham, UK: BMVA Press. http://citeseer.ist.psu.edu/article/cootes99comparing.html.
20. Basso C, Romdhani S, Blanz V, Vetter T. 2005. Morphable models of faces. In *Handbook of face recognition*, pp. 217–245. Ed S Li, A Jain. New York: Springer.
21. Blanz V, Vetter T. 2003. Face recognition based on fitting a 3D morphable model. *IEEE Trans Pattern Anal Machine Intell* **25**(9):1063- -1074.
22. Jones MJ, Poggio T. 1998. Multidimensional morphable models. *Proc IEEE Int Conf Comput Vision*, pp. 683–688. http://citeseer.ist.psu.edu/jones98multidimensional.html.
23. Bergen JR, Hingorani R. 1990. *Hierarchical motion-based frame rate conversion*. Technical Report, David Sarnoff Research Center, Princeton, NJ.
24. Li S, Zhang ZQ, Zhu L, Zhang HJ. 2002. Real-time multi-view face detection. In *Proc 5th Int Conf Automatic Face Gesture Recogn, Washington, DC, 20–21 May 2002*, pp. 149–154. Washington, DC: IEEE.
25. Schapire RE, Singer Y. 1999. Improved boosting algorithms using confidence-rated predictions. *Machine Learning* **37**(3):297–299.
26. Friedman J, Hastie T, Tibshirani R. 2000. Additive logistic regression: a statistical view of boosting. *Ann Stat* **28**(2):337–374.
27. Viola P, Jones M. 2001. Robust real-time object detection. *Int J Comput Vision*. To appear.
28. Friedman JH. 2001. Greedy function approximation: a gradient boosting machine. *Ann Stat*, **29**(5):1189–1232.
29. Mason L, Baxter J, Bartlett PL, Frean M. 1999. Functional gradient techniques for combining hypotheses. In *Advances in large margin classifiers*, pp. 221–247. Ed. AJ Smola, PL Bartlett, B Schölkopf, D Schuurmans. Cambridge: MIT Press.
30. Zemel RS, Pitassi T. 2001. A gradient-based boosting algorithm for regression problems. In *Advances in neural information processing systems*, Vol. 13. Ed. TK Leen, TG Dietterich, V Tresp. Cambridge: MIT Press. http://citeseer.ist.psu.edu/zemel01gradientbased.html.
31. Schapire RE, Freund Y, Bartlett P, Lee WS. 1998. Boosting the margin: a new explanation for the effectiveness of voting methods. *Ann Stat* **26**(5):1651–1686.
32. Freund Y, Schapire RE. 1997. A decision-theoretic generalization of on-line learning and an application to boosting. *J Comput Syst Sci* **55**(1):119–139.
33. Martinez AM, Benavente R. 1998. *The AR face database*. Computer Vision Center Technical Report No. 24. Barcelona, Spain.