Shape and Texture Based Deformable Models for Facial Image Analysis*

Stan Z. Li[1], Ying Zheng[2], Zhen Lei[1], ZengFu Wang[2]

[1] Center for Biometrics and Security Research &
National Laboratory of Pattern Recognition
Institute of Automation, Chinese Academy of Sciences, Beijing, China
[2] Department of Automation, Institute of Information Science and Technology
University of Science and Technology of China, Hefei, China
Contact: szli@nlpr.ia.ac.cn, http://www.cbsr.ia.ac.cn/users/∼szli

Abstract. In this chapter, we introduce concepts and algorithms of shape and texture based deformable models, more specifically Active Shape Models (ASM), Active Appearance models (AAM) and Morpahble Models, for 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 (DAM) and Texture Constrained ASM (TC-ASM), for improved fitting of shapes and textures. A method is also described for evaluation of goodness of fitting using 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 the system performance. For face analysis, in particular, both shapes and textures provide important clues useful for characterizing the faces.The task of face alignment is to accurately locate facial features such as the eyes, nose, mouth and outline, and normalize facial shape and texture. Accurate extraction and alignment of these features offer advantages for many applications.

A sort of the most successful face alignment methods is the deformable model, which can represent the variations in either shape or texture of the target objects. As two typical deformable models, the active shape models (ASM) [1] and active appearance models (AAM) [2, 3] have been widely used as alignment algorithms in medical image analysis and face analysis [4] for the past decade.

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

The standard ASM consists of two statistical models: (1) global shape model, which is derived from the landmarks in the object contour; (2) local appearance models, which is derived from the profiles perpendicular to the object contour around each landmark. ASM uses local models to find the candidate shape and the global model to constrain the searched shape. 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 *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. The minimization in high dimensional space is reduced in two models facilitate. This strategy is also developed in the active blob model[5].

ASM and AAM 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 result [8]. Blanz and Vetter extended morphable models and AAM to model relationship of 3D head geometry and facial appearance [9]. Li *et al.* [10] present a method for learning 3D face shape model from 2D images based on a shape-and-pose-free texture model. In Duta *et al.* [11], the shapes are automatically aligned using procrustes 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 are selected for each landmark to build local models. Baker and colleagues [13] propose an efficient method called "inverse compositional algorithm" for alignment. Ahlberg [14] extends AAM to a parametric method called Active Appearance algorithm to extract positions parameterized by 3D rotation, 2D translation, scale, and six Action Units (controlling the mouth and the eyebrows). In direct appearance model (DAM) [15, 16], shape is modeled as a linear function of texture. Using such an assumption, Yan *et al.* [17] propose texture-constrained ASM (TC-ASM), which has the advantage of ASM in having good localization accuracy and that of 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 Then we will 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 ASM and AAM. Then an alignment quality evaluation mechanism is addressed before the experimental results and conclusion to end this chapter.

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 Fig. 1. All the shapes are aligned or 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 $\overline{S}$. The warping may be done by pixel value interpolation, *e.g.* using a triangulation or thin plate spline method.

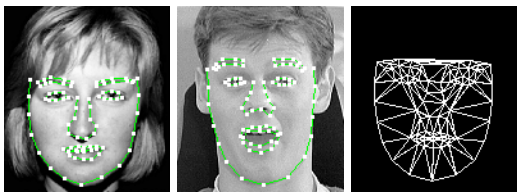

Fig. 1: Two face instaces labelled with 83 landmarks and the mesh of the mean shape. (Courtesy of S.C.Yan *et al.* [17])

2 Classical Deformable Models

There are two classical deformable models for 2D face analysis – Active Shape Model (ASM) and Active Appearance Model (AAM). We first look through them and 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 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 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 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 the 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:

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

where[3]

$$Eng(S_{lm}^n; s) = \lambda\|S_{lm}^n - S_{lm}^{n\,\prime}\|^2 + \|s_{lm}^n - s\|^2_{\mathbf{\Lambda}}. \tag{4}$$

In above equation, $s_{lm}^n = U^T(S_{lm}^n - \overline{S})$ is the projection of S_{lm}^n to the shape eigenspace, $S_{lm}^{n\,\prime} = \overline{S} + Us_{lm}^n$ is the reconstructed shape, $\mathbf{\Lambda}$ is the diagonal matrix of the largest eigenvalues of the training data $\{S_i\}$. The first term is the squared Euclidean distance from S_{lm}^n to the shape eigenspace, and the second is the squared Mahalanobis distance between s_{lm}^n and s. λ balances the two terms.

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

2.2 Active Appearance Model

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

[3] 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.

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 $\{T\}$, 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 modelled 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 large consumption of memory required by the learning of $\mathbf{A}_a$ and $\mathbf{A}_p$, the learning has to be done with a small, limited set of $\{\delta a, \delta T\}$.

2.3 3D Morphable Model

While ASM and AAM are for 2D image pattern analysis, here in this section, we temporarily deviate from the analysis of 2D face, and extend the dimension of face data to 3D by introducing morphable models [9, 22, 26]. 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 the 3D data processing system is the alignment between the input data and the standard one. The 3D alignment may involve many rigid or non-rigid transformations. For 3D face analysis,in particular, the 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 spirits of multidimensional morphable model[20] and 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 learnt from a database of 3D faces, *i.e.*, faces acquired by a 3D *Cyberware*^{TM} scanner.Building a 3DMM requires to transform 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 vector space. In [21], 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 then can be generated by convex combination of the K exemplar faces, with its shape and texture vectors, S and T, are:

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

Again, PCA is applied separately on the shape and texture space to reduce the 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 matrix 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, 3DMM can recover the vectors of s and t by an *analysis by synthesis* fashion, providing an alignment between input face and exemplar faces in database.

3 Motivations for Improvements

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

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

First, let us look into 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 can not 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.

Then, let us 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 Eq.(7) and (8) the formula
$$\mathbf{W}a = \begin{pmatrix} \mathbf{\Lambda}s \\ t \end{pmatrix} \tag{15}$$
when that s contains some components which are independent of t.
4. However, in 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 been 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 of 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 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 DAM is the memory consumption: the regression of $\mathbf{A}_a$ in AAM is very memory consuming. AAM prediction needs to model linear

relationship between appearance and 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 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 Direct Appearance Model (DAM), for aligning and estimating face appearances.

The new appearance model is motivated by our findings of a flaw of AAM modeling and difficulties in training 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, which is composed of both shape and texture, to be determinable by using just the corresponding texture. 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 AAM's crucial idea of modeling the AAM appearance subspace from shape and texture combined. In this way, DAM includes admissible textures previously unseen by AAM, and improves the convergence and accuracy.

Another merit of 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 in AAM. This cuts down the memory requirement to a large extent, and further improves the convergence and accuracy.The claimed advantages of 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 in 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 the 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 $\delta\mathbf{R}$

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

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

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

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 the optimal solution

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

The minimized cost is the trace of the following

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

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

$$\delta p = \mathbf{R}_p \delta T' \tag{22}$$

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

$$\delta T' = \mathbf{H}^T \delta T \tag{23}$$

$\delta T'$ 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 be 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 is 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 DAM needs only about 1/20 of memory required by AAM. For 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 training data for 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 training data for $\mathbf{A}_a$ is $200 \times 113 \times 4 = 90,400$. Therefore, the size of training data for AAM's prediction matrices is $90,400 + 4,800 = 95,200$, which is 19.83 times that for 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 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 models 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 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 the 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 Fig.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 Fig.3.

Fig. 2: Frontal, half-side, and full-side view faces and the labeled landmark points.(Courtesy of S.Z.Li *et al.* [16])

The left side models and right side models are reflections of each other, so we only need to train one side of them. 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 view in $[-90°, 90°]$.

The multi-view DAM search has a similar process to that of 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 the 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 Eq.(22) and (23).The search algorithm is summarized below:

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

Table 1: Dimensionalities of shape and texture variations for face data. #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 ($\delta T'$).(Courtesy of S.Z.Li *et al.* [16])

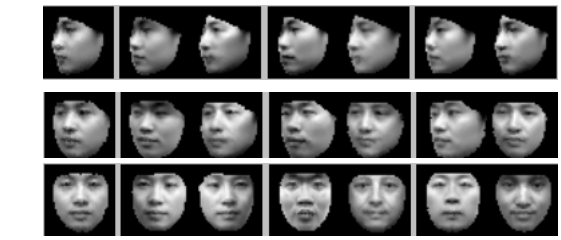

Fig. 3: Texture and shape variations due to variations in the first three principal components of the texture (The shapes change in accordance with $s = \mathbf{R}t$) for full-side ($\pm 1\sigma$), half-side ($\pm 2\sigma$), and frontal ($\pm 3\sigma$) views. (Courtesy of S.Z.Li*et al.* [16])

1. Initialize the position parameters p_0, and determine 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 the texture subspace $\mathbb{S}_t$ as t, reconstruct the 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 the 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 the 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 [28], as shown in Fig.4. A multi-resolution pyramid structure is used in search to improve the result. Fig.5 demonstrates scenarios of how DAM converges.

When the face is undergone large variation due to stretch in either the x or y direction, the model fitting can be improved by allowing different scales in the two directions. This is done by splitting the scale parameter into two: s_x and s_y. improvement is demonstrated in Figs.6 on Page 14.

Fig. 4: Initial alignment provided by a multi-view face detector. (Courtesy of S.Z.Li *et al.* [16])

Fig. 5: DAM aligned faces (from left to right) at the 0-th, 5-th, 10-th, and 15-th iterations, and the original images for (top-bottom) frontal, half-side and full-side view faces.(Courtesy of S.Z.Li *et al.* [16])

5 Texture Constrained Active Shape Model

TC-ASM [17] imposes the linear relationship of direct appearance model (DAM) to improve ASM search. The motivation is the following: ASM has better accuracy in shape localization than 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, 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 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 is orthogonal to $\mathbb{S}_t$ (projected back to $\mathbb{R}^L$) and $\dim(\mathbb{S}_t) \ll \dim(T)$, the dimension of the space $\{\delta T\}$ is very high, and it is hard to train the regression matrix $\mathbf{A}_a, \mathbf{A}_p$ and the prediction of the variance of position can be subject to

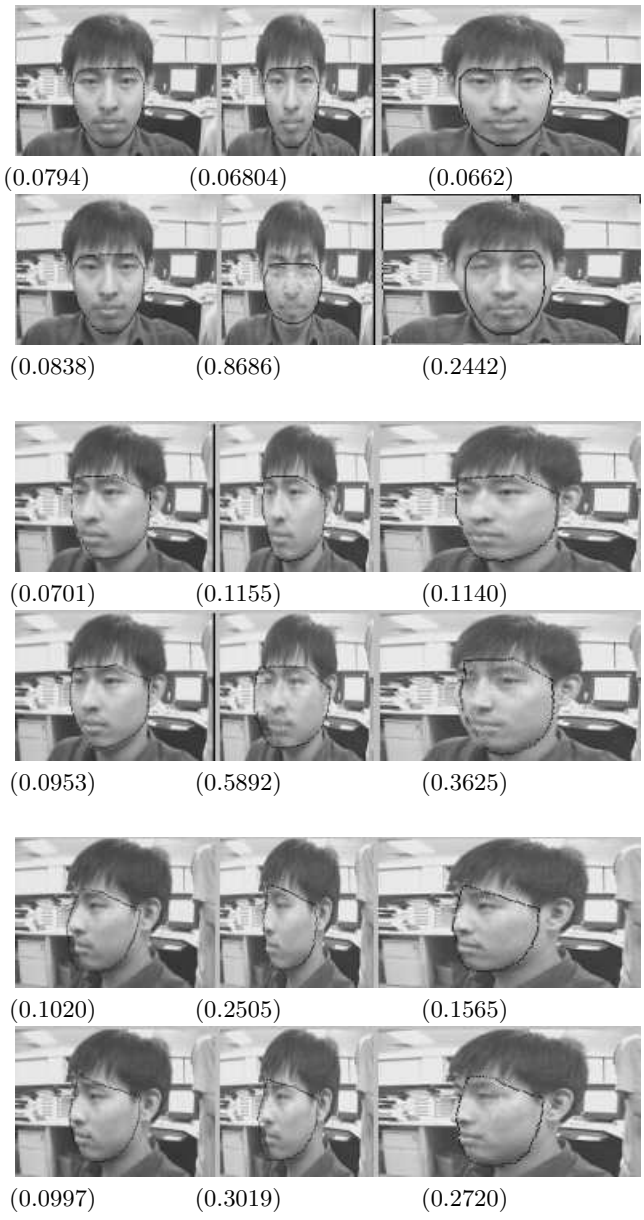

Fig. 6: Results of non-isometric (top of each of the three blocks) and isometric (bottom) search for frontal (top block), half-side (middle block) and full-side (bottom block) view faces. From left to right of each row are normal, and stretched faces. The number below each result is the corresponding residual error. (Courtesy of S.Z.Li *et al.* [16])

significant errors. Also it is time and memory consuming. TC-ASM is aimed to overcome the above problems.

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 ASM and AAM. The texture-constrained shape model, or the mapping from texture to shape, is simply assumed linear and could be easily learnt. In each step of the optimization, a better shape is found under 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, whilst 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 the 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 the training pairs $\{(s_i, t_i)\}$ by singular-value decomposition. For simplicity, $\mathbf{\Sigma}_t$ is assumed to be a known constant matrix. Fig.7 demonstrates the accuracy of the prediction in the test data via the 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.

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

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

where the energy function is:

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

5.2 TC-ASM in Bayesian Framework

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

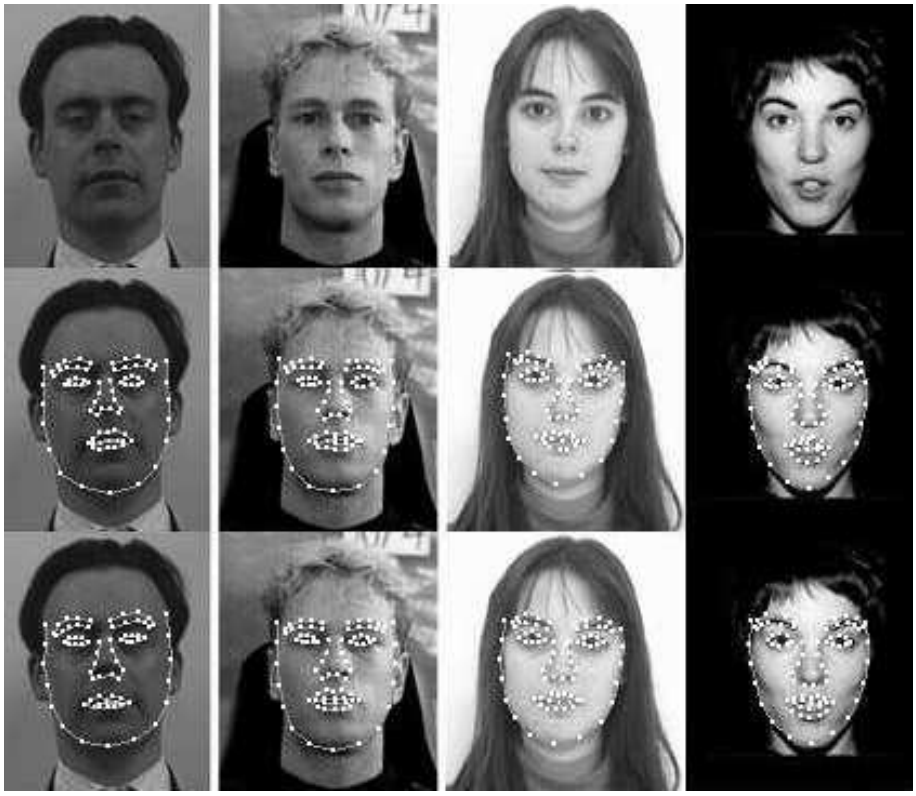

Fig. 7: The comparison of the manually labeled shape (middle row) and the shape (bottom row) derived from the enclosed texture using the learned projection matrix: $s_t = \mathbf{R}t$. In the top row are the original images. All the images are test data. (Courtesy of S.C.Yan *et al.* [17])

1. Set the iteration number $n = 1$;
2. Using the local appearance models in ASM, we may obtain the candidate shape S_{lm}^n with the shape parameters s_{lm}^n based on the shape $S^{(n-1)}$ of the previous iteration;
3. The texture enclosed by S_{lm}^n is warped to the mean shape, denoted by t^n. The texture-constrained shape s_t^n is predicted from t^n by Eq.(25);
4. The *p*osterior (MAP) estimation of S^n or s^n given S_{lm}^n and s_t^n 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 the step 4 and the stopping condition in detail. To simplify the notation, we shall omit the superscript n in following deduction since the iteration number is constant. In step 4, the *p*osterior (MAP) estimation

of s given 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 to 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 the Eq.(4) and Eq.(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

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

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, TC-ASM could be regarded as a trade-off between 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 in AAM, the optimal solution must have been touched. In practice, however, these two shapes would hardly turn to be the same. A threshold is introduced to evaluate the similarity and sometimes the convergence criterion in 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 optimization process.

6 The Evaluation for Face Alignment

The emergence of many effective face alignment algorithms serves as a contrast to the lack of an effective method for the evaluation of face alignment results. In 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 AAM and DAM, the PCA reconstruction error is used as a distance measure for the

evaluation of alignment quality.However, the reconstruction error may not be a good discriminant for the evaluation of alignment quality because a non-face can look like a face when projected onto the PCA face subspace. In TC-ASM, the algorithm claims to reach a convergence when the solution generated by ASM is verified in AAM, whereas both convergence criterions 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 un-qualified alignment results. The positive subset consists of qualified face alignment examples and the negative subset consists of obviously un-qualified and near-but-not-qualified examples.

We use AdaBoost algorithm [29, 30] for the 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 significant better results than other features such as Haar wavelets. The 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 is 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 [31]. Such a divide-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 an evaluation functions thus learned and the linear evaluation function of reconstruction error used in AAM. First, the evaluation is learned in such a way to distinguish between good and bad alignment. Secondly, 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 un-qualified 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 progressing, more subtle adjustments are made. The result can gives a good match to the target image or it can fail (see Figure. 8). The 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.

Similar problem exists in AAM search. There, the PCA reconstruction error is used as a distance measure for the evaluation of alignment quality (and for

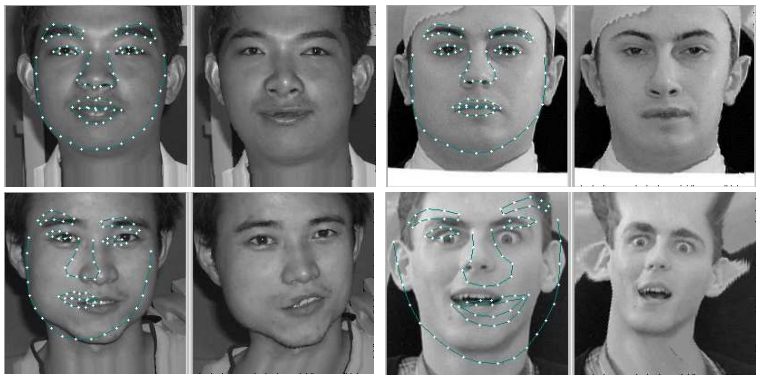

Fig. 8: Four face instances of qualified (top) and un-qualified (bottom) examples with their warped images. (Courtesy of Huang *et al.* [18])

guiding the search as well). However, the reconstruction error may not be a good discriminant for the 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 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 un-qualified 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 mean shape, as shown in Fig. 8. Face alignment quality evaluation can be posed as a two class classification problem: given an alignment result x (*i.e.* warped face), the 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 labelled 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 [29, 30], 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 the 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 which are erroneously classified previously. It is noted in recent studies [32–34] that the artificial operation of explicit re-weighting is unnecessary and can be incorporated into a functional optimization procedure of boosting.

---

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)
   $H(x) = \text{sign}[\sum_{m=1}^{M} h_m(x)]$.

Fig. 9: RealBoost Algorithm. (Courtesy of Huang *et al.* [18])

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 as a measure of the confidence of the h's prediction. The upper bound on classification error achieved by H_M can be derived as the following exponential loss function [35]

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

AdaBoost construct $h_m(x)$ 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 which leads to the minimum cost

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

The minimizer is [29, 30]

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

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 one if C is true or zero otherwise. $P(y=-1|x,w^{(M-1)})$ is defined similarly.

The AdaBoost algorithm based on the descriptions from [29, 30] is shown in Fig. 9. There, the re-weight formula in step 2.(3) is equivalent to the multiplicative rule in the original form of AdaBoost [36, 29]. 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 modelling 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 the 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 grey-scale image. In this work, we use the 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:

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

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

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

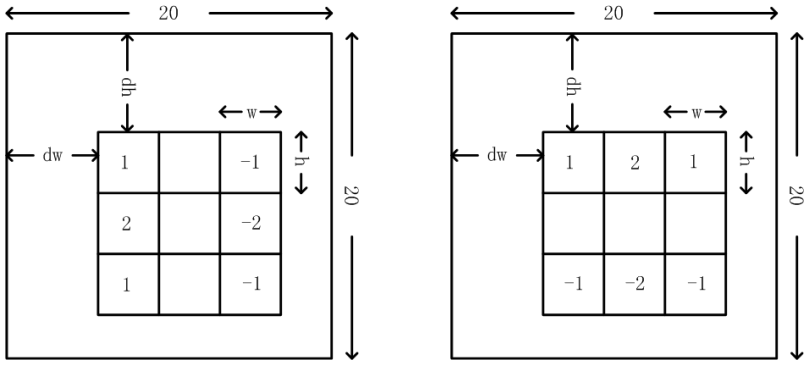

Fig. 10: The two types of simple Sobel-like filters defined on sub-windows. The rectangles are of size $w \times h$ and are at distances of (dw, dh) apart. Each feature takes a value calculated by the weighted $(\pm 1, \pm 2)$ sum of the pixels in the rectangles. (Courtesy of Huang *et al.* [18])

6.4 Statistical Learning of Weak Classifiers

A scalar feature $z'_k : x \rightarrow \mathbf{R}$ is a transform from the n-dimensional (400-D if a face example x is of size 20x20) data space to the real line. These block differences are an extension to the Sobel filters. For each face example of size 20x20, 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 LLR's for the target $L_M(x)$:

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

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

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

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

Computation of Subspaces A total of 80 images of size 128x128 are collected. Each image contains a different face in an area of about 64x64 pixels. The images set is 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 Fig. 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 AAM and DAM.

For 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 which retains 98% of total variation of the concatenated features.

For 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}.

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

DAM requires much less memory during the learning of the prediction matrices $\mathbf{R}_p$ in Eq.(22) than AAM for learning $\mathbf{A}_a$ in Eq.(11). For 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 the training $\mathbf{R}_p$. So, the size of training data for DAM is $80 \times 4 \times 6 = 1920$. For 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 training data for $\mathbf{A}_a$ is

Fig. 11: A face image and the landmark points. (Courtesy of X.W.Hou *et al.* [15])

$80 \times 65 \times 4 = 20800$. Therefore, the size of training data for AAM's prediction matrices is $20800 + 1920 = 22720$, which is 11.83 times that for DAM. On a PC, for example, the memory capacity for AAM training with 80 images would allow DAM training with 946 images.

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

	$E(\|\delta T\|^2)$	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%

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). (Courtesy of X.W.Hou *et al.* [15])

Fig. 12 illustrates average scenarios of DAM and AAM alignment. Fig. 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 DAM has faster convergence and smaller error than AAM.

Multi-View DAM The training set contains 200 frontal, 200 half-side, and 170 full-side view faces whose sizes are of about 64x64 pixels, while the test set contains 80 images for each view group. The landmark points are labeled manually (see Fig.2 and Table 1). They are used for the training and as ground-truth in the test stage.

To compare, we also implemented 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 which retains 98% of the total variation of the concatenated features.

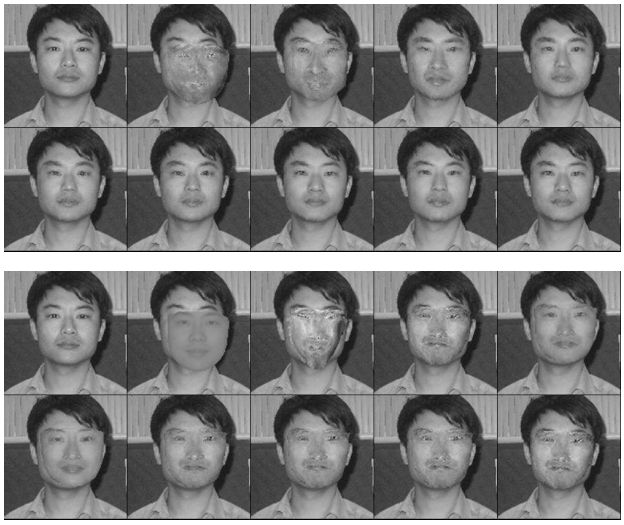

Fig. 12: Scenarios of DAM (top) and AAM (bottom) divergence (Courtesy of X.W.Hou *et al.* [15])

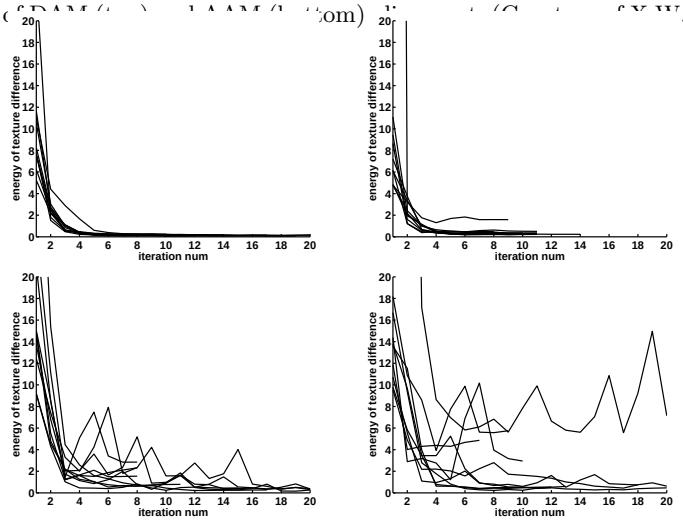

Fig. 13: The evolution of total δT for the DAM (top) and AAM (bottom) as a function of iteration number for the training (left) and test (right) images. (Courtesy of X.W.Hou *et al.* [15])

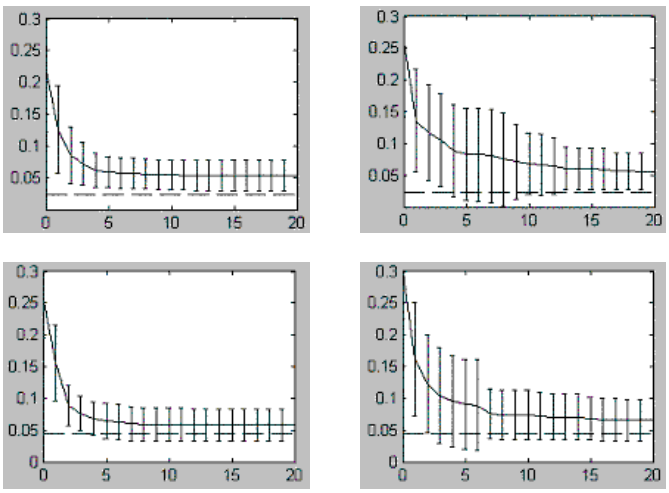

Fig. 14: Mean error (the curve) and standard deviation (the bars) in reconstructed texture $\|\delta T\|$ as a function of iteration number for the DAM (left) and AAM (right) methods with the training (top) and test (bottom) sets, for frontal face images. The horizontal dashed lines in the lower part of the figures indicate the average $\|\delta T\|$ for the manually labeled alignment. (Courtesy of S.Z.Li *et al.* [16])

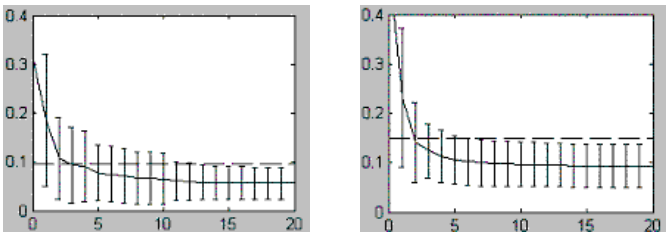

Fig. 15: Mean error in $\|\delta T\|$ and standard deviation of the DAM alignment for half- (left) and full- (right) side view face images from the test set. Note that the mean errors in the calculated solutions are smaller than obtained using the manually labeled alignment after a few iterations. (Courtesy of S.Z.Li *et al.* [16])

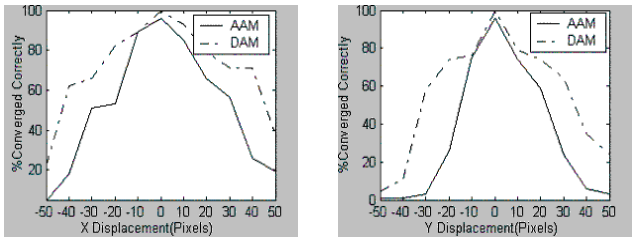

Fig. 16: Alignment accuracy of DAM (dashed) and AAM (solid) in terms of localization errors in the x (left) and y (right) directions. (Courtesy of S.Z.Li *et al.* [16])

Some results about DAM learning and search have been presented in Fig.2 – Fig.6. Fig.14 compares the convergence rate and accuracy properties of 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 the 80 images from test set. We see that DAM has faster convergence rate and smaller error than AAM. Fig.15 illustrates the error of DAM for non-frontal faces. Fig.16 compares the alignment accuracy of 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 additional 20 other test images. It shows again that DAM is more accurate than 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 320x240 pixels. Every view model takes about 10 iterations to converge. If 3 view models are searched with per face, as is done with image sequences from video, the algorithm takes about 1 second to find the best face alignment.

7.2 TC-ASM

A data set containing 700 face images with different illumination conditions and expressions are selected from the AR database[37] in our experiments, each of which is 512×512, 256 grays image 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, 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 TC-ASM. By means of PCA with 98% total variations retained, the dimension of the shape parameter in ASM shape space is reduced to 88, and the texture parameter vector in AAM texture space is reduced to 393. The concatenated vector of the shape and texture parameter vector with the weighting parameter $\gamma = 13.77$ is reduced to 277. Two types of experiments are presented: (1) the comparison of the point-position accuracy and (2) the comparison of the texture reconstruction error. The experiments are all performed in the 3-level resolution image pyramid.

Point Position Accuracy The average point-point distances between the searched shape and the manually labeled shape of the three models are compared in Fig.17. The vertical axis represents the percentage of the solutions for which the average point-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 ASM and AAM in most cases since the curve of TC-ASM lies above the curves for ASM and AAM. It also suggests that AAM outperforms ASM when the initial displacement is small, while ASM is more robust to the increasing of the initial displacement.

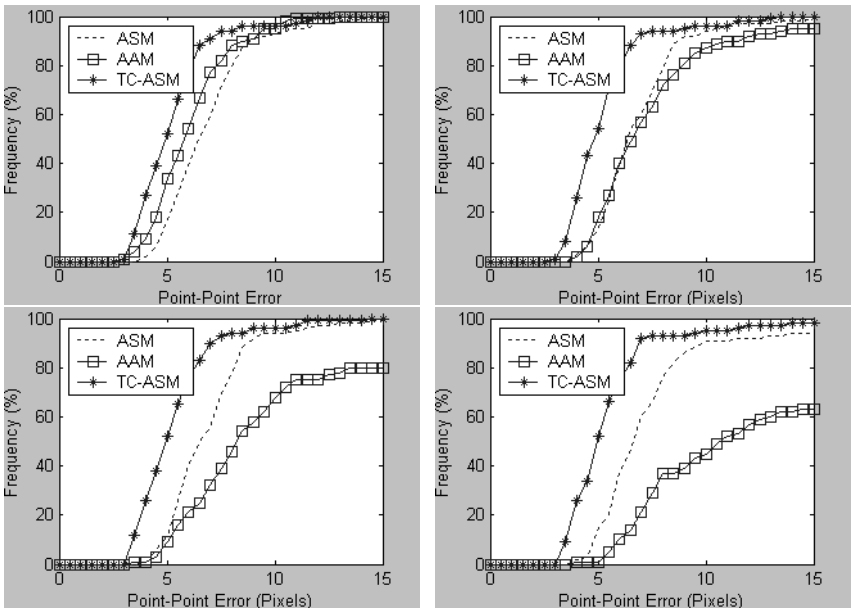

Fig. 17: Accuracy of ASM, AAM, TC-ASM. From upper to lower, left to right are the results obtained with the initial displacements of 10, 20, 30 and 40 pixels. Note that the value of vertical coordinate is the percentage of examples that have the point-point distance smaller than the corresponding value of horizonal coordinate. (Courtesy of S.C.Yan *et al.* [17])

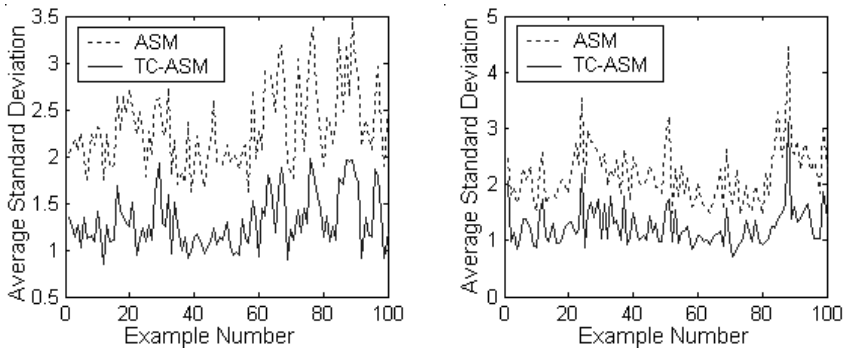

Fig. 18: Standard deviation in the results of each example for ASM (dotted) and TC-ASM (solid) with training set (left) and test set(right).(Courtesy of S.C.Yan *et al.* [17])

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

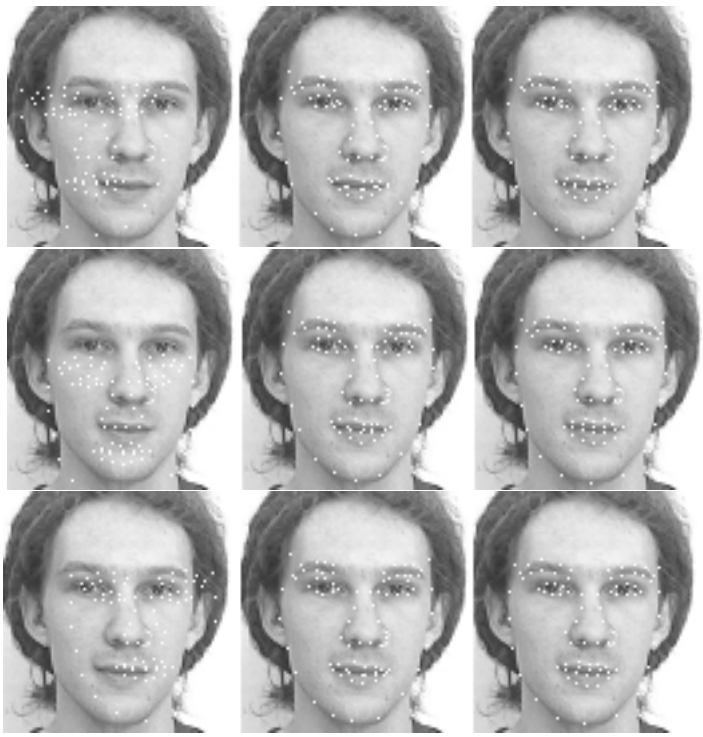

Fig. 19: Stability of ASM (Middle column) and TC-ASM (Right column) in shape localization. The different initialization conditions are showed in the left column.(Courtesy of S.C.Yan *et al.* [17])

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

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

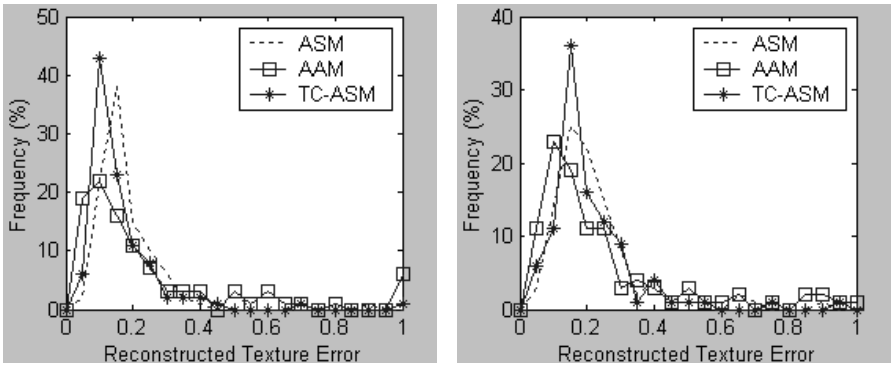

Fig. 20: Distribution of the texture reconstruction error in ASM(dotted), AAM(square) and TC-ASM(asterisk) with training data (left) and test data (right). (Courtesy of S.C.Yan *et al.* [17])

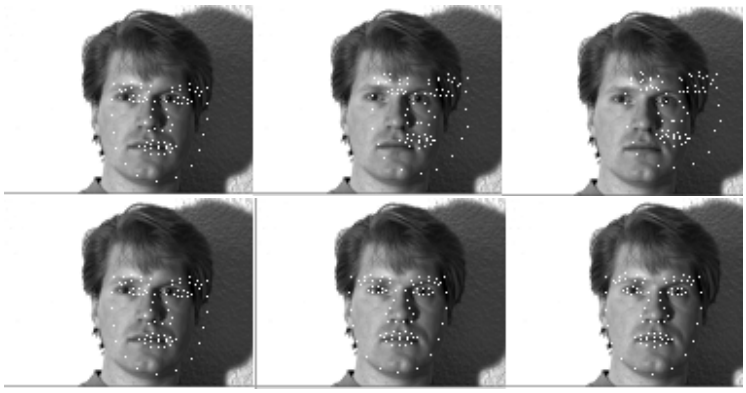

Fig. 21: Sensitivities of AAM (upper) and TC-ASM (lower) to illumination condition not seen in the training data. From left to right are the results obtained at the 0-th, 2-th, and 10-th iterations.(Result in different level of image pyramid is scaled back to the original scale) (Courtesy of S.C.Yan *et al.* [17])

From the experiment, TC-ASM is more computationally expensive than ASM, but it is much faster than AAM. In our experiment (600 training images, 83 landmarks and a P-III 667 computer with 256M memory), it takes averagely 32 ms per iteration, which is twice of ASM (16 ms) but one fifth of AAM (172

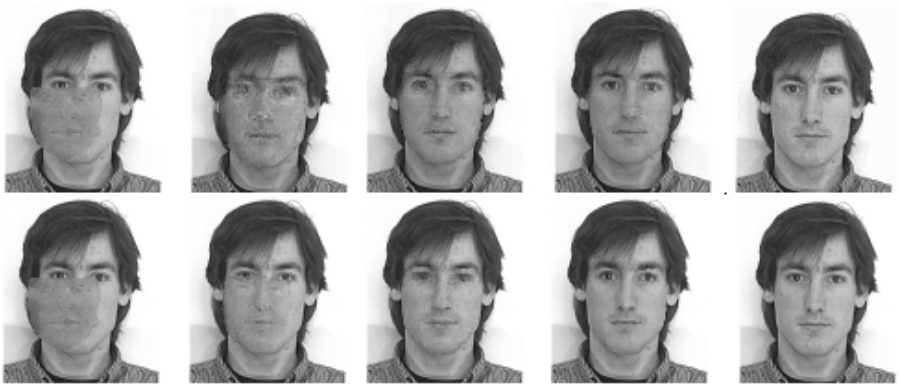

Fig. 22: Scenarios of AAM (upper) and TC-ASM (lower) alignment with texture reconstruct error 0.3405 and 0.1827 respectively. From left to right are results obtained at the 0-th, 5-th, 10-th, 15-th iterations and the original image. (Result in different level of image pyramid is scaled back to the original scale) (Courtesy of S.C.Yan *et al.* [17])

ms). The training time of AAM is more than two hours, while TC-ASM is only about 12 minutes.

7.3 The Evaluation for Face Alignment

The positive and negative training and set date are generated as follows: All the shapes are aligned or 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, shifting positive examples' shape points. A strong classifier is trained to reject 92% negative examples, while correctly accepting 100% of positive examples.

A cascade of classifiers is trained to train a computational effective model, makes training easier with divide-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. 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 need 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. Fig. 23 shows the ROC curve for the reconstruction error based alignment eval-

uation method for the training set. Note that this method cannot achieve 100% correct rates.

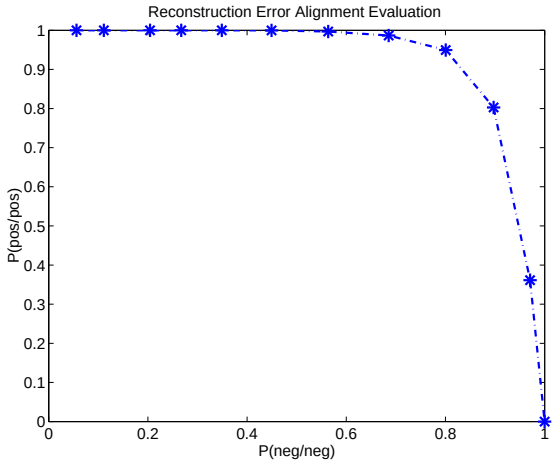

Fig. 23: ROC curve for the reconstruction error based alignment evaluation for the training set. (Courtesy of Huang *et al.* [18])

During the test, a total of 1528 aligned examples (800 qualified images 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 dist_{PCA} − threshold for the PCA based method. The qualified and un-qualified alignment decision is judged by comparing the score with the normalized threshold of 0. Some examples of accepted (the top part) and rejected (the bottom part) face alignment results are shown in Fig.24. Fig. 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) means the false positive rate and so on.

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

Table 3: Training results (WC: weak classifier) (Courtesy of Huang *et al.* [18])

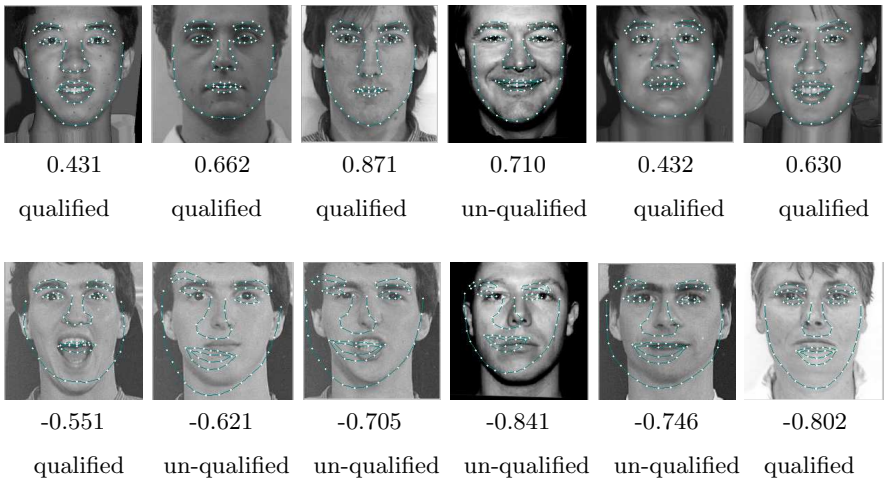

Fig. 24: Alignment quality evaluation results: accepted images (top) and rejected images (bottom) (Courtesy of Huang *et al.* [18])

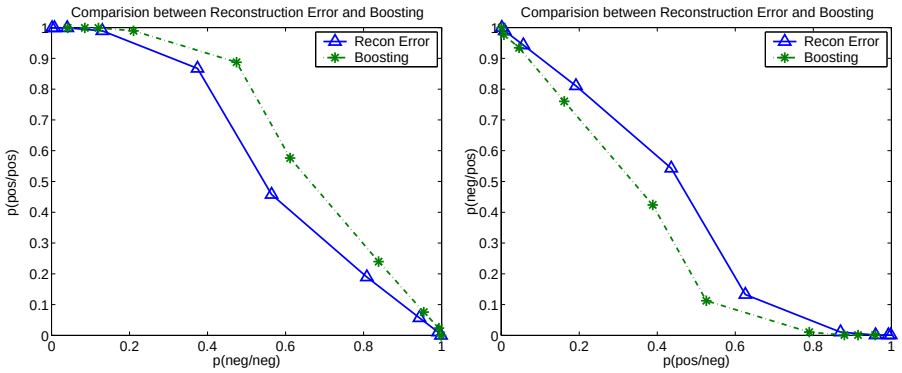

Fig. 25: Comparision between reconstruction error method and boost method (Courtesy of Huang *et al.* [18])

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

8 Conclusion

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

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

References

1. Cootes, T.F., Taylor, C.J., Cooper, D.H., Graham, J.: Active shape models: Their training and application. CVGIP: Image Understanding **61** (1995) 38–59
2. Cootes, T.F., Edwards, G.J., Taylor, C.J.: Active appearance models. In: ECCV98. Volume 2. (1998) 484–498
3. Edwards, G.J., Cootes, T.F., Taylor, C.J.: Face recognition using active appearance models. In: Proceedings of the European Conference on Computer Vision. Volume 2. (1998) 581–695
4. Cootes, T.F., , Taylor, C.J.: Statistical models of appearance for computer vision. Technical report, `www.isbe.man.ac.uk/~bim/refs.html` (2001)
5. Sclaroff, S., Isidoro, J.: Active blobs. In: Proceedings of IEEE International Conference on Computer Vision, Bombay, India (1998)
6. Cootes, T.F., Walker, K.N., Taylor, C.J.: View-based active appearance models. In: Proc. Int. Conf. on Face and Gesture Recognition. (2000) 227–232
7. S. Romdhani, A.P., Gong, S.: Learning a single active face shape model across views. In: Proc. IEEE International Workshop on Recognition, Analysis, and Tracking of Faces and Gestures in Real-Time Systems, Corfu, Greece (1999)
8. Cootes, T., Taylor, C.: Constrained active appearance models. Proceedings of IEEE International Conference on Computer Vision **1** (2001) 748–754
9. Blanz, V., T.Vetter: A morphable model for the synthesis of 3d faces. In: SIGGRAPH'99 Conference Proceedings. (1999) 187–194
10. Li, Y., Gong, S., Liddell, H.: Constructing facial identity surfaces in a nonlinear discriminating space. In: Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition, Hawaii (2001)
11. Duta, N., Jain, A., Dubuisson-Jolly, M.: Automatic construction of 2-d shape models. IEEE Transactions on Pattern Analysis and Machine Intelligence **23** (2001) 433–446
12. van Ginneken, B., Frangi, A.F., Staal, J.J., ter Haar Romeny, B.M., Viergever, M.A.: A non-linear gray-level appearance model improves active shape model segmentation. In Proceedings of Mathematical Methods in Biomedical Image Analysis (2001)

13. Baker, S., Matthews, I.: Equivalence and efficiency of image alignment algorithms. In: Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition. Volume 1., Hawaii (2001) 1090–1097
14. Ahlberg, J.: 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 (2001) 68–72
15. Hou, X.W., Li, S.Z., Zhang, H.J.: Direct appearance models. In: Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition. Volume 1., Hawaii (2001) 828–833
16. Li, S.Z., Yan, S.C., Zhang, H.J., Cheng, Q.S.: Multi-view face alignment using direct appearance models. In: Proceedings of IEEE International Conference on Automatic Face and Gesture Recognition, Washington, DC (2002)
17. Yan, S.C., Liu, C., Li, S.Z., Zhu, L., Zhang, H.J., Shum, H., Cheng, Q.: Texture-constrained active shape models. In: Proceedings of the First International Workshop on Generative-Model-Based Vision (with ECCV), Copenhagen, Denmark (2002)
18. Huang, X., Li, S.Z., Wang, Y.: Statistical learning of evaluation function for asm/aam image alignment. In: Workshop on Biometric Authentication, Prague (2004)
19. Cootes, T.F., Edwards, G.J., Taylor, C.J.: Comparing active shape models with active appearance models. In: British Machine Vision Conference. (1999)
20. Jones, M.J., Poggio, T.: Multidimensional morphable models. In: Proc. of the International Conference on Computer Vision. (1998) 683–688
21. Bergen, J.R., Hingorani, R.: Hierarchical motion-based frame rate conversion. Technical Report (1990)
22. S. Romdhani, V. Blanz, C.B., Vetter, T.: Morphable Models of Faces. In: Handbook of Face Recognition. Springer (2004)
23. Romdhani, S., Vetter, T.: Efficient, robust and accurate fitting of a 3d morphable model. In: Proceedings of the IEEE International Conference on Computer Vision 2003. (2003)
24. S. Romdhani, V.B., Vetter, T.: Face identification by fitting a 3d morphable model using linear shape and texture error functions. In: Computer Vision-ECCV. (2002)
25. Romdhani, S., Vetter, T.: Estimating 3d shape and texture using pixel intensity, edges, specular highlights, texture constraints and a prior. In: IEEE Conference on Computer Vision and Pattern Recognition. (2005)
26. Blanz, V., T.Vetter: Face recognition based on fitting a 3d morphable model. IEEE Transactions on Pattern Analysis and Machine Intelligence **25** (2003) 1063–1074
27. V. Blanz, C. Basso, T.P., T.Vetter: Reanimating faces in images and video. In: Proceedings of EUROGRAPHICS. (2003)
28. Z. Q. Zhang, L. Zhu, S.L., Zhang, H.J.: Real-time multi-view face detection. In: Proceedings of IEEE International Conference on Automatic Face and Gesture Recognition, Wanshington,DC (2002)
29. Schapire, R.E., Singer, Y.: Improved boosting algorithms using confidence-rated predictions. In: Proceedings of the Eleventh Annual Conference on Computational Learning Theory. (1998) 80–91
30. Friedman, J., Hastie, T., Tibshirani, R.: Additive logistic regression: a statistical view of boosting. The Annals of Statistics **28** (2000) 337–374
31. Viola, P., Jones, M.: Robust real time object detection. In: IEEE ICCV Workshop on Statistical and Computational Theories of Vision, Vancouver, Canada (2001)
32. Friedman, J.: Greedy function approximation: A gradient boosting machine. The Annals of Statistics **29** (2001)
33. Mason, L., Baxter, J., Bartlett, P., Frean, M.: Functional gradient techniques for combining hypotheses. In Smola, A., Bartlett, P., Schölkopf, B., Schuurmans, D., eds.: Advances in Large Margin Classifiers. MIT Press, Cambridge, MA (1999) 221–247
34. Zemel, R., Pitassi, T.: A gradient-based boosting algorithm for regression problems. In: Advances in Neural Information Processing Systems. Volume 13., Cambridge, MA, MIT Press (2001)
35. Schapire, R., Freund, Y., Bartlett, P., Lee, W.S.: Boosting the margin: A new explanation for the effectiveness of voting methods. Annals of Statistics **26** (1998) 1651–1686

36. Freund, Y., Schapire, R.: A decision-theoretic generalization of on-line learning and an application to boosting. Journal of Computer and System Sciences **55** (1997) 119–139
37. Martinez, A., Benavente, R.: The AR face database. Technical Report 24, CVC (1998)