Active Shape Models,

Modeling Shape Variations and Gray Level Information
and an Application to Image Search and Classification

Ghassan Hamarneh
The Imaging and Image Analysis Group
Department of Signals and Systems
Chalmers University of Technology
Gothenburg, Sweden.

mailto:jessi@ae.chalmers.se
http://www.ae.chalmers.se/~jessi/

Abstract

This report presents a review and an investigation of a shape modeling method referred to as Active Shape Modeling. The appearance of objects in images varies due to several reasons including lighting effects, 3D pose, and other. Here we represent objects by a number of landmark points and then model the shape variations of an object using principal component analysis to obtain a point distribution model. We also model the gray level appearance of objects. We then use our knowledge about the appearance of objects in image search and classification of objects.

Acknowledgement

This investigation was funded by the Swedish Foundation for Strategic Research as part of the national research programme VISIT (VISual Information Technology).

Table of Contents

Index Terms

Shape Modeling, Shape Variation, Point Distribution Model, Image Search, Gray Level Appearance, Image Classification, Active Shape Models.

Variables and Notation

n	*number of landmarks*
N	*number of training images or shapes*
j	*index for landmarks*
I, k	*index for images or shapes*
$\mathcal{S}$	*training set of images*
l, m	*index for principal components or their weights*
s	*scaling factor*
θ	*rotation angle in radians*
t_x	*translation in the x direction*
t_y	*translation in the y direction*
$\mathbf{t}$	*translation vector*
L	*number of levels in the image pyramid*
q	*index for levels in the image pyramid*

1. Introduction

The purpose of this report is to review and investigate a method referred to as Active Shape Models (ASM). ASM was originally proposed by Cootes *et al.* [1] The modeling technique is similar in spirit to Active Contour Models, or snakes, proposed by Kass *et al.* [2], but the advantage of ASM is that instances of models can only deform in ways found in a training set. That is, they allow for considerable variability but are still specific to the class of objects or structures they intend to represent.

Images of the same object can appear in a variety of ways due to several reasons, as will be discussed in section 2.1. In Section 2.2 we introduce the need to model such shape variations. In 2.3 we represent the shapes of objects by a set of landmark points. In the remaining part of section 2 we work our way up to modeling the shape variations of a set of training images. An overview and the detailed steps of modeling the shape variations are presented in sections 2.4 and 2.5, respectively. We will see that in order to obtain such a model we need to align the set of shapes in the training set. Section 3 explains how we use the shape variation model and the gray level information to search in an image for an instance of an object. Modeling the gray level information in the training set of images is discussed in section 4. Section 5 is devoted to explaining the iterative procedure of the image search. Section 6 discusses how image search can be improved by performing a multi-resolution image search. In Section 7, we discuss the possibility of using such models with additional gray level information to perform image classification. Three appendices follow, the first describes how to obtain a weighting matrix that reflects the stability of landmark points within a training set of images. The second solves the problem of aligning shapes. The third discusses how the principal components of an observation matrix can be obtained.

2. Point Distribution Model

2.1. Shape variations

Many examples of objects can be found that posses the characteristic of having a shape that varies. The human heart, face, and hand, resistors appearing on a circuit board (Cootes *et al.* [1]), and many others, are all objects that can appear in different images as one of a variety of possible shapes. The human heart can be found in any instant of its beating cycle, the human face can have many expressions, the hand can be in one of infinitely many different gestures, and resistors are not manufactured identically. In addition, all of the above shapes can be of different objects and thus having different appearances, such as two different humans showing the same expression. Other causes of variations in shapes - besides the individual appearance and facial expressions - are lighting effects, 3D pose (Lanitis *et al.* [3]), and highly probable is the combination of two or more of such causes (see Figure 1)[4].

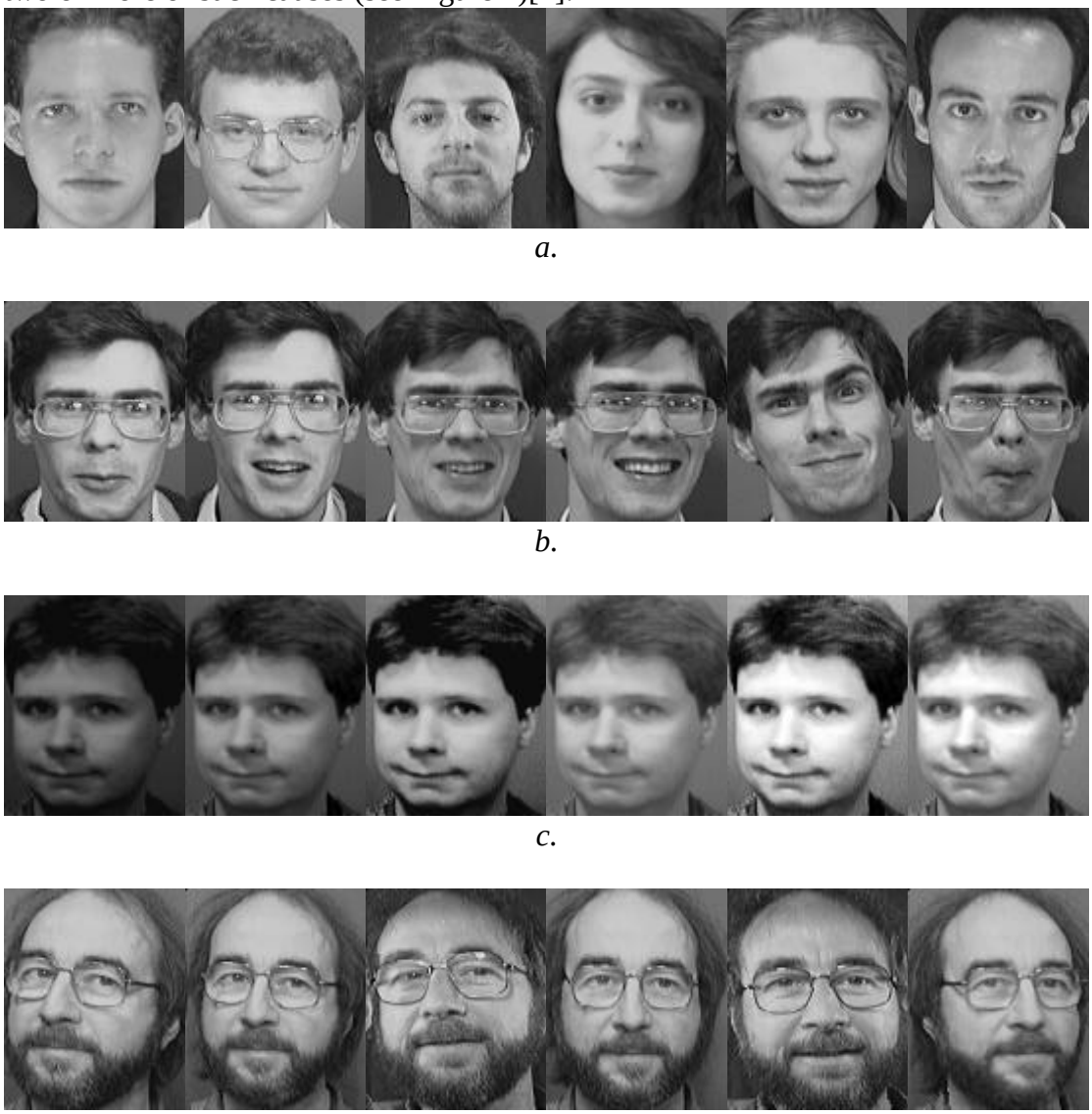


Figure 1. variations due to: a. individual appearance, b. facial expressions, c. lighting effects, and d. 3D pose

2.2. Modeling Shape Variations

The aim is to build a model that describes shapes and typical variations of an object. To make the model able of capturing typical variations, we collect different images of that object, and aim that the object is appearing in different ways reflecting its possible variations. This set of images is named the training set (see Figure 2) [4].

Figure 2. samples from a training set

2.3. Pre-modeling Shape Representation

To collect information about the shape variations needed to build the model, we represent each shape with a set of landmark points or landmarks. The number of landmarks should be adequate to show the overall shape and also show details where it is needed. Each shape will then be represented by a predefined number of landmarks, this number depends on the complexity of the object's shape and the desired level of detail description (see Figure 3).

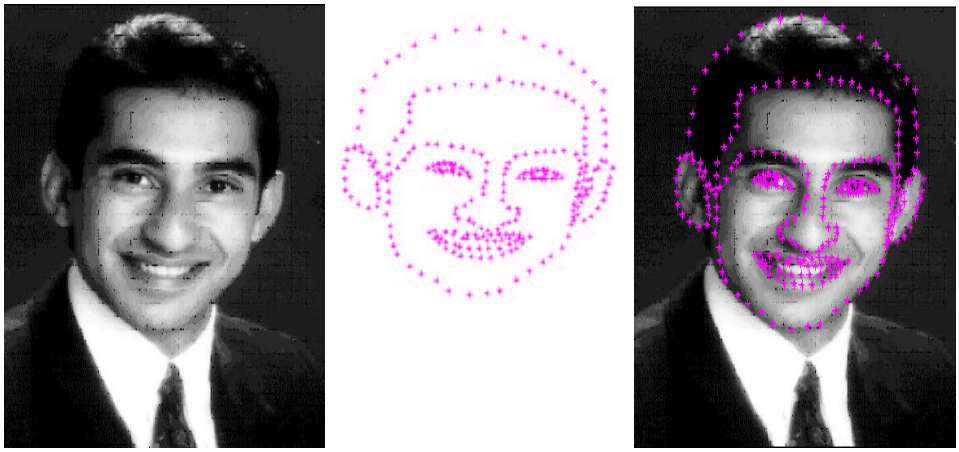


Figure 3. an image, the landmarks, and the image with the landmarks

2.4. Point Distribution Model - Overview

The model that will be used to describe a shape and its typical variations is based on the variations of the position of each landmark point within the training set. Each landmark point will thus have a certain distribution in the image space. We want to model the variation distribution of the landmark points and hence the name Point Distribution Model (PDM). To obtain the PDM, as we said earlier, we need a training set of images containing the object in different forms, and then we need to represent the shape in each image by labeling the set of landmark points. A step of aligning the shapes (represented now by their landmarks) in the different images of the training set

will be needed for proper description of the variation. In the final step of obtaining the PDM we need to study the variations of the landmarks and summarize it in a compact form. The steps are described in more details in the following sub-section.

2.5. Point Distribution Model - Steps

Here we will present the steps needed to obtain the PDM for an object found in typical forms of its variation in a set of training images. The first step is to label all the shapes in the training set, then align the labeled shapes, and finally capture the statistics of the variations. The training set is denoted $\mathcal{S}$, which contains N images.

2.5.1. Labeling the training set

Before starting labeling the shapes in the training set, we should decide on the number of landmark points that adequately represent the shape. Let us denote the number of landmarks by n. Then for each image of the training set, we locate the shape (by eye), and then identify the landmarks on that shape. The identification of the landmark points in the shapes is called labeling the training set. It is important to correctly and accurately specify the landmark points. There should be exact correspondence in the order of labeling a shape in one image and in another. The location of the landmark point should be located as accurately as possible, since these locations will govern the resulting point variations and the intended PDM.

There are some basic principals that suggest where to choose the landmark points of a shape (Cootes *et al.* [1]):

1. Application-dependent landmarks.
2. Application-independent landmarks.
3. Landmarks interpolated from the above two.

Examples of the application-dependent landmarks are the center of the eye in a model of the face, or the apex point in a model of the heart. Application-independent landmarks such as the highest point of an object in a certain orientation. Interpolated landmarks such as points separated by equal distances, along a certain path, between two points of type 1 and 2. Typically, type 3 will dominate and describe most of the boundary of the shape.

As a result, we end up with n landmark coordinate points for each of the N images. Put in another way, we end up with N landmark coordinate points for each landmark of the shape. We donate the j^{th} landmark coordinate point of the i^{th} image of the training set by (x_{ij}, y_{ij}), and the vector describing the n points of the i^{th} shape in the training set by

$$\mathbf{x}_i = \left[x_{i0}, y_{i0}, x_{i1}, y_{i1}, \ldots, x_{in-1}, y_{in-1}\right]^T \text{ where } 1 \le i \le N \qquad (1)$$

2.5.2. Aligning the training set

As a result of labeling the training set we have a set of N vectors, $\mathbf{x}_i$, $1 \le i \le N$. In order to study the variations of the position of each landmark throughout the set of training images, all the shapes (each represented by its corresponding landmarks vector $\mathbf{x}$) must be aligned to each other. The aligning is done by changing the pose (scaling, rotating, and translating) of shapes as necessary to obtain the alignment.

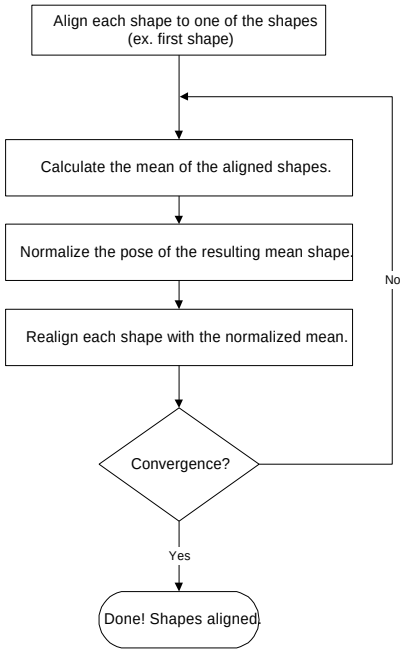
2.5.2.1. Aligning two shapes

We first introduce the problem of aligning two shapes or two landmark vectors. A weighted least-squares approach is used to obtain the pose (scaling, rotating, and translating) parameters needed to align one shape or vector to the other. This approach is explained in [Appendix 2]. Expressed mathematically, given two vectors $\mathbf{x}_i$ and $\mathbf{x}_k$, we are required to align $\mathbf{x}_k$ to $\mathbf{x}_i$, so we need to find the scaling value: s, the rotation angle: θ, and the value of translation in both dimensions: (t_x, t_y), performed on the vector $\mathbf{x}_k$ and needed to align it to $\mathbf{x}_i$ in a weighted least-squares sense.

A weighting matrix is used to give more significance to the landmark points that tend to be more stable. The stability of a point is measured by having less movement relative to the other points. [Appendix 1] explains how to obtain the weighting matrix.

2.5.2.2. Aligning all shapes

The following algorithm is used for aligning the set of N shapes to each other (Cootes *et al.* [1]):



The pose of a shape is described by its scaling, rotation, and translation, with respect to a known reference.

Normalization of the pose means a) *scaling* of the shape so that the distance between two points becomes a certain constant, b) *rotating* the shape so that the line joining two pre-specified landmarks is directed in a certain direction, and c) *translating* the shape so that it becomes centered at a certain coordinate. Normalization is carried out in order to force the process to converge, otherwise the mean shape may translate or expand (or shrink) indefinitely.
Convergence is established if the shapes are not changing more than a pre-defined threshold.

2.5.3. Capturing the statistics

The i^{th} aligned shape of the training set of images is represented by the vector $\mathbf{x}_i$, where $\mathbf{x}_i$, now, contains the new coordinates resulting from aligning. This vector is of dimension *2n*, so it can be represented by a point in a *2n*-dimensional space. The *N* vectors representing the *N* aligned shapes will then map to a 'cloud' of *N* points in the same *2n*-D space. It is assumed that these *N* points are contained within a region of this *2n*-D space, this region is called the 'Allowable Shape Domain' (ASD). Every point in this region gives a shape that is similar to the other shapes in this ASD. We can use the following as a measure of similarity: The less the Euclidean distance between two points representing two shapes in the *2n*-D the more similar the shapes are. The Euclidean distance *d* between the two points representing the two shapes $\mathbf{x}_i$ and $\mathbf{x}_k$ is given by:

$$d_{ik} = \sqrt{(x_{i0} - x_{k0})^2 + (y_{i0} - y_{k0})^2 + (x_{i1} - x_{k1})^2 + (y_{i1} - y_{k1})^2 + \ldots + (x_{in-1} - x_{in-1})^2 + (y_{in-1} - y_{in-1})^2}$$

or

$$d_{ik}^2 = (\mathbf{x}_i - \mathbf{x}_k)^T (\mathbf{x}_i - \mathbf{x}_k) \qquad (2)$$

or with a possible weighting matrix $\mathbf{W}$ that may be used to give more importance to those landmark points that vary less with in the training set (see [Appendix 2]), we have.

$$d_{ik} = \sqrt{\begin{array}{l} w_o(x_{i0} - x_{k0})^2 + w_o(y_{i0} - y_{k0})^2 + w_1(x_{i1} - x_{k1})^2 + w_1(y_{i1} - y_{k1})^2 + \\ \ldots + w_{n-1}(x_{in-1} - x_{kn-1})^2 + w_{n-1}(y_{in-1} - y_{kn-1})^2 \end{array}}$$

or

$$d_{ik}^2 = (\mathbf{x}_i - \mathbf{x}_k)^T \mathbf{W} (\mathbf{x}_i - \mathbf{x}_k) \qquad (3)$$

where $\mathbf{x}_i = [x_{i0}, y_{i0}, x_{i1}, y_{i1}, \ldots, x_{ij}, y_{ij}, \ldots x_{in-1}, y_{in-1}]^T$

and $\mathbf{W} = diag(w_o, w_o, w_1, w_1, \ldots, w_j, w_j, \ldots w_{n-1}, w_{n-1})$

Now, we wish to find the driving principals governing the behavior of the variations of the *N* points in the *2n*-D space defined by the *2n* variables of $\mathbf{x}$. Applying Principal Component Analysis (PCA) [Appendix 3], we generate a new set of variables called the principal components. Each principal component is a linear combination of the original variables (referred to as a Standardized Linear Combination in Mardia & Bibby [5]). All the principal components are orthogonal to each other so there is no redundant information. The principal components as a whole form an orthogonal basis for the space of data (Bradley [6]).

The first principal component is a single axis in space. When projecting each of the observations (*N* vectors representing the shapes) on that axis, the resulting values form a new variable. The variance of this variable is the maximum along all possible choices of the first axis, *i.e.* represents the maximum shape variation.

The second component is another axis in space, perpendicular to the first. Projecting our *N* observations on this axis generates another new variable. The variance of this variable is the maximum among all possible choices of this second axis. The dimension

of both the full set of principal components and the original set of variables is the same hence both sets contain *2n* variables.

In many applications it can be assumed that the first few principal components describe a high percentage of the total variance of the original data. Hence, the dimension of the model can be reduced and the variations described by a less number of variables (less than *2n*), that is performing what is referred to as "parsimonious summarization" (Bradley [6], Mardia & Bradley [7]).

Now, we can express each landmark vector as a linear combination of the principal components. Moreover, we can express the difference between each vector and the mean of all vectors as a linear combination of the principal components, because this difference vector will also lie in the *2n*-D space spanned by the principal components.

Denoting the mean vector by $\bar{\mathbf{x}}$, and the difference vector between the vector $\mathbf{x}_i$ and $\bar{\mathbf{x}}$ by $d\mathbf{x}_i$, we have

$$d\mathbf{x}_i = \mathbf{x}_i - \bar{\mathbf{x}} \qquad (4)$$

and

$$\bar{\mathbf{x}} = \frac{1}{N}\sum_{i=1}^{N}\mathbf{x}_i \qquad (5)$$

The covariance matrix for the landmarks of the shapes is given as

$$\mathbf{C}_\mathbf{x} = \frac{1}{N}\sum_{i=1}^{N}(\mathbf{x}_i - \bar{\mathbf{x}})(\mathbf{x}_i - \bar{\mathbf{x}})^T \qquad (6)$$

Representing the difference $d\mathbf{x}_i$ as a linear combination of the principal components, we write

$$d\mathbf{x}_i = b_{i0}\mathbf{p}_0 + b_{i1}\mathbf{p}_1 + \ldots + b_{i2n-1}\mathbf{p}_{2n-1} \qquad (7)$$

where $\mathbf{p}_l$ is the l^{th} principal component axis or column vector, normalized to unit length, *i.e.* $\mathbf{p}_l{}^T\mathbf{p}_l = 1$, and because the principal components are also mutually orthogonal, then they are orthonormal.

$$\mathbf{p}_l{}^T\mathbf{p}_m = \begin{Bmatrix} 1 & l = m \\ 0 & l \neq m \end{Bmatrix} \qquad (8)$$

b_{il} is a scalar that weighs $\mathbf{p}_l$ for the i^{th} shape.

Equivalently, we can write

$$\mathbf{x}_i = \bar{\mathbf{x}} + d\mathbf{x}_i \qquad (9)$$

and rewrite $d\mathbf{x}_i$ as

$$d\mathbf{x}_i = b_{i0}\mathbf{p}_0 + b_{i1}\mathbf{p}_1 + \ldots + b_{i2n-1}\mathbf{p}_{2n-1} = \mathbf{P}\mathbf{b}_i \qquad (10)$$

where $\mathbf{b}_i = \begin{bmatrix} b_{i0} & b_{i1} & \ldots & b_{i2n-1} \end{bmatrix}^T$, and $\mathbf{P} = \begin{bmatrix} \mathbf{p}_0 & \mathbf{p}_1 & \ldots & \mathbf{p}_{2n-1} \end{bmatrix}$

this yields

$$\mathbf{x}_i = \bar{\mathbf{x}} + \mathbf{P}\mathbf{b}_i \qquad (11)$$

$\mathbf{b}_i$ can be found as

$$\mathbf{b}_i = \mathbf{P}^{-1}(\mathbf{x}_i - \bar{\mathbf{x}}) \qquad (12)$$

With $\mathbf{P}$ being an orthogonal matrix, a square matrix with orthonormal columns, (see Strang [8, page 167]), we have $\mathbf{P}^T\mathbf{P} = \mathbf{P}\mathbf{P}^T = \mathbf{I}$ so $\mathbf{P}^{-1} = \mathbf{P}^T$, and $\mathbf{b}_i$ is written as

$$\mathbf{b}_i = \mathbf{P}^T(\mathbf{x}_i - \bar{\mathbf{x}}) \qquad (13)$$

To summarize, we have N shape-representing vectors having a mean shape $\bar{\mathbf{x}}$, each vector can be expressed as the sum of the mean shape and a weighted sum of the principal components.

If we choose the weights as $\mathbf{b}_i = \mathbf{P}^T(\mathbf{x}_i - \bar{\mathbf{x}})$ with $1 \le i \le N$, we will end up with the shape $\mathbf{x}_i$, but we might as well choose other weights, such as $\mathbf{b} = \mathbf{P}^T(\mathbf{x} - \bar{\mathbf{x}})$ where $\mathbf{x} \notin \{\mathbf{x}_1 \quad \mathbf{x}_2 \quad \ldots \quad \mathbf{x}_N\}$, and end up with a shape not being in the training set, and how similar this shape is to the ones in the training set is determined by the distance (d_{ik}) between it and them. If we want the vector of weights $\mathbf{b}$ to be chosen so that the resulting shape can be considered an acceptable or allowable shape of the object, then certain constraints on the these weights must be put. If we constrain b_l to:

$$b_{l\,\min} \le b_l \le b_{l\,\max}; \text{ for } 0 \le l \le 2n-1 \qquad (14)$$

then it is suitable to choose

$$b_{l\,\min} = -b_{l\,\max} \qquad (15)$$

and $b_{l\,\max}$ proportional to the variance of the training set's projection along the l^{th} principal component. Finding the values of the variances is explained in [Appendix 3].

The only thing remaining is to *find* the principal components. In [Appendix 3] it is explained, in general, how to find the principal components for an a observation matrix containing *m* rows of observations and *n* columns of variables. In our case we have *N* observations (shape-representing vectors) and 2*n* variables (the (x, y) coordinates for the *n* landmarks of a shape).

Recalling our aim of this analysis, which is to reduce the dimensionality of the data and describe the variations with a fewer number of variables, we now express the *N* shape-representing vectors having a mean shape $\bar{\mathbf{x}}$, as the sum of the mean shape and a weighted sum of *not all* the principal components. We assume that the first *t* (out of *2n*) principal components explain a sufficiently high percentage of the total variance of the original data. By this we are saying that the *2n*-D cloud of shapes has a 'small' width in the direction of the $t+1^{st}$ principal component and the ones that follow. Our basic equation becomes,

$$\mathbf{x} = \bar{\mathbf{x}} + \mathbf{P}\mathbf{b} \qquad (16)$$

where $\bar{\mathbf{x}} = \frac{1}{N}\sum_{i=1}^{N}\mathbf{x}_i$ and considering only *t* principal components, we have

$\mathbf{b} = \begin{bmatrix} b_0 & b_1 & \ldots & b_{t-1} \end{bmatrix}^T$ and $\mathbf{P} = \begin{bmatrix} \mathbf{p}_0 & \mathbf{p}_1 & \ldots & \mathbf{p}_{t-1} \end{bmatrix}$

and the constraints on $\mathbf{b}$ become

$$b_{l\,\min} \le b_l \le b_{l\,\max}, \text{ for } 0 \le l \le t-1 \qquad (17)$$

3. Using PDM for Image Search

Having a training set of images containing a certain object in many possible variations, we labeled the training set, aligned the shapes and obtained a Point Distribution Model. In addition, we assume that *t* principal components describe a sufficiently large percentage of the data variations. We further more assume that for a shape to be allowable the scalars weighing the principal components should lie within a certain range.

Now, we deal with the problem of locating an instance of an object in an image. The basic idea is to start with an initial estimate. Then we examine the neighborhood of the landmarks of our estimate aiming to find better locations of the landmarks. We then change the shape and the pose of our estimate to better fit the new locations of the landmarks while producing in the process a new acceptable or allowable shape. The fact that the shapes are modeled in a way that they can only vary in a controlled way (by constraining the weights of the principal components) explains why such models are named Active Shape models or ASMs.

We assume that an instance of an object is described as the sum of the mean shape obtained from the training and a weighted sum of the principal components, with the possibility of this sum being translated, rotated, or scaled.

3.1. The initial estimate

Denoting the operation of scaling and rotating the shape-representing vector $\mathbf{x}$ by s and q, respectively, by $M(s,q)\mathbf{x}$, and letting $\mathbf{t}_i = \begin{bmatrix} t_{xi} & t_{yi} & t_{xi} & t_{xi} & \dots & t_{xi} & t_{xi} \end{bmatrix}^T$ with a length of *2n*. We can express the initial estimate $\mathbf{x}_i$ of the shape as a scaled, rotated and translated version of a shape $\mathbf{x}_I$:

$$\mathbf{x}_i = M(s_i,q_i)[\mathbf{x}_I] + \mathbf{t}_i \qquad (18)$$

$\mathbf{x}_I$ can also be expressed as $\mathbf{x}_I = \bar{\mathbf{x}} + d\mathbf{x}_I$, with $d\mathbf{x}_I = \mathbf{P}\mathbf{b}_I$, or $\mathbf{x}_I = \bar{\mathbf{x}} + \mathbf{P}\mathbf{b}_I$ then we have

$$\mathbf{x}_i = M(s_i,q_i)[\bar{\mathbf{x}} + d\mathbf{x}_I] + \mathbf{t}_i \qquad (19)$$

as our initial estimate.

3.2. The suggested movements

Now examining the region surrounding each landmark point of $\mathbf{x}_i$ we find that $\mathbf{x}_i$ should move to $\mathbf{x}_i + d\mathbf{x}_i$. Adjustments could be made along the normal of the shape boundary towards the strongest image edge, with a magnitude proportional to the strength of the edge. Alternative methods also exist, and we will discuss that later. For now, we assume that the adjustments are known to be $d\mathbf{x}_i$.

3.3. Computing the shape and pose parameters

So we need to adjust the pose (scaling, rotation and translation) parameters, as well as the shape parameters (the weights of the principal components) to move our current estimate $\mathbf{x}_i$ as close as possible to $\mathbf{x}_i + d\mathbf{x}_i$, while still satisfying the shape constraints imposed to produce an acceptable or allowable shape. Put symbolically,

$$\mathbf{x}_i = M(s_i, q_i)[\mathbf{x}_I] + \mathbf{t}_i \xrightarrow{(1+ds),dq,d\mathbf{t}} \mathbf{x}_i + d\mathbf{x}_i$$

or

$$M(s_i(1+ds), q_i + dq)[\mathbf{x}_I] + \mathbf{t}_i + d\mathbf{t} \rightarrow \mathbf{x}_i + d\mathbf{x}_i \qquad (20)$$

We first find the additional scaling $1+ds$, rotation dq and translation (dt_x, dt_y), required to move $\mathbf{x}_i$ as close as possible to $\mathbf{x}_i + d\mathbf{x}_i$ (as explained in [Appendix 2]), then there will remain residual adjustments which can only be satisfied by deforming the shape $\mathbf{x}_I$. We wish to calculate the adjustments $d\mathbf{x}$ (to be added to $\mathbf{x}_I$), needed to cause $\mathbf{x}_i$ to move to $\mathbf{x}_i + d\mathbf{x}_i$ when combined with effect of the new scaling, rotation, and translation parameters (Cootes *et al.* [1]).

Knowing $1+ds$, dq and $d\mathbf{t}$, we need to solve the following equation for $d\mathbf{x}$:

$$M(s_i(1+ds), q_i + dq)[\mathbf{x}_I + d\mathbf{x}] + \mathbf{t}_i + d\mathbf{t} = \mathbf{x}_i + d\mathbf{x}_i$$

or

$$M(s_i(1+ds), q_i + dq)[\mathbf{x}_I + d\mathbf{x}] = \mathbf{x}_i + d\mathbf{x}_i - (\mathbf{t}_i + d\mathbf{t}) \qquad (21)$$

using

$$\mathbf{x}_i = M(s_i, q_i)[\mathbf{x}_I] + \mathbf{t}_i$$

we get:

$$M(s_i(1+ds), q_i + dq)[\mathbf{x}_I + d\mathbf{x}] = M(s_i, q_i)[\mathbf{x}_I] + d\mathbf{x}_i - d\mathbf{t} \qquad (22)$$

since

$$M^{-1}(s,q)[\ldots] = M(s^{-1}, -q)[\ldots] \qquad (23)$$

we have

$$\mathbf{x}_I + d\mathbf{x} = M((s_i(1+ds))^{-1}, -(q_i + dq))[M(s_i, q_i)[\mathbf{x}_I] + d\mathbf{x}_i - d\mathbf{t}] \qquad (24)$$

which gives

$$d\mathbf{x} = M((s_i(1+ds))^{-1}, -(q_i + dq))[M(s_i, q_i)[\mathbf{x}_I] + d\mathbf{x}_i - d\mathbf{t}] - \mathbf{x} \qquad (25)$$

In general, the resulting vector $d\mathbf{x}$ is in $2n$-D space, but since there are only t (less than $2n$) modes of variation described in our model, we can only move the shape in t dimensions described by the first t principal axes. So we seek the vector that is most similar to $d\mathbf{x}$ but lies in the t-D space. If we adopt the least-squares approach, then the solution $d\mathbf{x}'$ is the projection of $d\mathbf{x}$ onto the t-D space (the space spanned by the vectors of principal components, or the t columns of $\mathbf{P}$). Put in equations (see Lantis *et al.* [3]) we have:

$$d\mathbf{x}' = \mathbf{A}d\mathbf{x} \qquad (26)$$

where $\mathbf{A}$ is a projection matrix given as (Strang [8, page 170])

$$\mathbf{A} = \mathbf{P}(\mathbf{P}^T\mathbf{P})^{-1}\mathbf{P}^T \qquad (27)$$

Since the columns of $\mathbf{P}$ are orthonormal and $\mathbf{P}$ is no longer square, we have $\mathbf{P}^T\mathbf{P} = \mathbf{I}$ and thus

$$d\mathbf{x}' = \mathbf{P}\mathbf{P}^T d\mathbf{x} \qquad (28)$$

So, instead of $\mathbf{x}_I$ moving to $\mathbf{x}_I + d\mathbf{x}$ it will move $\mathbf{x}_I + d\mathbf{x}'$. Expressing $d\mathbf{x}'$ as

$$d\mathbf{x}' = \mathbf{P}d\mathbf{b}' \qquad (29)$$

and multiplying by $\mathbf{P}^T$ from the left, we get

$$d\mathbf{b}' = \mathbf{P}^T d\mathbf{x}' \qquad (30)$$

3.4. Updating the shape and pose parameters

With this information, we are ready to update the shape and pose parameters of our initial estimate. We obtain a new estimate $\mathbf{x}_i^{(1)}$ where

$$\mathbf{x}_i^{(1)} = M\left(s_i(1+ds), q_i + dq\right)\left[\mathbf{x}_I + \mathbf{P}d\mathbf{b}'\right] + \mathbf{t}_i + d\mathbf{t} \qquad (31)$$

We then start with $\mathbf{x}_i^{(1)}$ the same we started with $\mathbf{x}_i$ and produce $\mathbf{x}_i^{(2)}$, and so on, until no significance change in the shape is noticeable.

Put in another way, the parameters are updated as follows,

$$t_{xi} \longrightarrow t_{xi} + dt_x$$

$$t_{yi} \longrightarrow t_{yi} + dt_y$$

$$s_i \longrightarrow s_i(1+ds)$$

$$q_i \longrightarrow q_i + dq$$

$$\mathbf{b}_l \longrightarrow \mathbf{b}_l + d\mathbf{b}' \qquad (32)$$

Weights can be added, as follows

$$t_{xi} \longrightarrow t_{xi} + w_t dt_x$$

$$t_{yi} \longrightarrow t_{yi} + w_t dt_y$$

$$s_i \longrightarrow s_i(1+w_s ds)$$

$$q_i \longrightarrow q_i + w_q dq$$

$$\mathbf{b}_l \longrightarrow \mathbf{b}_l + \mathbf{W}_b d\mathbf{b}' \qquad (33)$$

where w_t, w_s, w_q are scalars. $\mathbf{W}_b$ is diagonal matrix of weights possibly the identity matrix, but alternatively, having weights proportional to the standard deviation of the corresponding shape parameter over the training set. We should always remember to limit the adjustments of $\mathbf{b}$ to such that it gives an acceptable or allowable shape.

To summarize,

- We start with an initial shape $\mathbf{x}_i = M\left(s_i, q_i\right)\left[\mathbf{x}_I\right] + \mathbf{t}_i$.
  (where $\mathbf{x}_I = \bar{\mathbf{x}} + \mathbf{P}\mathbf{b}_I$, $\mathbf{P}$ contains only the first *t* principal components, and $\bar{\mathbf{x}}$ is the mean shape).

- We find the required movements $d\mathbf{x}_i$ that move the landmarks to a better location in the image.
  (discussed later on).

- We find the additional pose parameter changes $(1+ds, dq, d\mathbf{t})$ required to move $\mathbf{x}_i$ as close as possible to $\mathbf{x}_i + d\mathbf{x}_i$.
  (explained in [Appendix 2]).

- We find the additional shape parameter changes $d\mathbf{b}'$.
  (using $d\mathbf{b}' = \mathbf{P}^T d\mathbf{x}'$, where $d\mathbf{x}' = \mathbf{P}\mathbf{P}^T d\mathbf{x}$ and
  $d\mathbf{x} = M\left((s_i(1+ds))^{-1}, -(q_i + dq)\right)\left[M(s_i, q_i)[\mathbf{x}_I] + d\mathbf{x}_i - d\mathbf{t}\right] - \mathbf{x}$)
- We update the shape and pose parameters acting on $\mathbf{x}_I$ and with the resulting new shape we can start the process again.
- The procedure is repeated until no considerable change is observed.

4. Modeling the gray level Appearance

We pointed out earlier the need to find the desired movement of the landmark points of the current estimate to new better locations during image search. Here we will discuss how we can make use of the gray level (not only shape) information in the training set of images to find such suggested movements.

The idea is that we examine the gray level information around each landmark in all the training set of images. We then try to put such information in a compact form and use it while performing image search. This gray level information is used to cause the landmark points of the estimate to move to better locations when performing image search.

In general, any region around the landmarks can be considered and studied, but here we will concentrate on the gray levels only along a line passing through the landmark in question and perpendicular to the boundary formed by the landmark and its neighbors.

For every landmark point j in the image i of the training set, we extract a gray level profile $\mathbf{g}_{ij}$, of length n_p pixels, centered at the landmark point. We do not work with the actual gray level profile, but we work with the normalized derivative, this gives invariance to the offsets and uniform scaling of the gray levels (Cootes *et al.* [9]).

The gray level profile of the landmark j in the image i is a vector of n_p values,

$$\mathbf{g}_{ij} = \begin{bmatrix} g_{ij0} & g_{ij1} & \dots & g_{ijn_p-1} \end{bmatrix}^T \qquad (34)$$

and the derivative profile of length $n_p - 1$ becomes

$$d\mathbf{g}_{ij} = \begin{bmatrix} g_{ij1} - g_{ij0} & g_{ij2} - g_{ij1} & \dots & g_{ijn_p-1} - g_{ijn_p-2} \end{bmatrix} \qquad (35)$$

The normalized derivative profile is given by

$$\mathbf{y}_{ij} = \frac{d\mathbf{g}_{ij}}{\sum_{k=0}^{n_p-2} \left| d\mathbf{g}_{ijk} \right|} \qquad (36)$$

Now, we calculate the mean of the normalized derivative profiles of each landmark throughout the training set, and we get for landmark j

$$\bar{\mathbf{y}}_j = \frac{1}{N} \sum_{i=1}^{N} d\mathbf{g}_{ij} \qquad (37)$$

The covariance matrix of the normalized derivative is given by

$$\mathbf{C}_{\mathbf{y}_j} = \frac{1}{N} \sum_{i=1}^{N} (\mathbf{y}_{ij} - \bar{\mathbf{y}}_j)(\mathbf{y}_{ij} - \bar{\mathbf{y}}_j)^T \qquad (38)$$

We have modeled the gray level information, and now we have for each landmark a certain profile $\bar{\mathbf{y}}_j$, that we can use for the image search, and particularly finding the desired movements of the landmarks that take $\mathbf{x}_i$ to $\mathbf{x}_i + d\mathbf{x}_i$.

5. Finding the desired movement

Here will use the information obtained from modeling the gray level statistics around each landmark to obtain the desired movement or adjustment of each landmark ($d\mathbf{x}_i$), during image search that will take each landmark to a better location in order to better fit the image.

To find such adjustments, we search along a line passing through the landmark in question and perpendicular to the boundary formed by the landmark and its neighbors, thus we obtain a search profile. Within this search profile, we are looking for a sub profile with characteristics that match the ones obtained from training. In order to do so, we collect the gray level values along the search profile, then take the derivative and normalize, the same way we did in the training process, to obtain the normalized derivative profile. We then search within the normalized derivative search profile for a sub-profile that matches the *mean* normalized derivative profile obtained earlier.

The search profile along the landmark j is denoted $\mathbf{s}_j$ whose length is n_s (surely $n_s > n_p$)

$$\mathbf{s}_j = \begin{bmatrix} s_{j0} & s_{j1} & \dots & s_{j,n_s-2} & s_{j,n_s-1} \end{bmatrix}^T \quad (39)$$

the derivative search profile of landmark i will be of length $n_s - 1$ as follows

$$d\mathbf{s}_j = \begin{bmatrix} s_{j1} - s_{j0} & s_{j2} - s_{j1} & \dots & s_{j,n_s-2} - s_{j,n_s-3} & s_{j,n_s-1} - s_{j,n_s-2} \end{bmatrix}^T \quad (40)$$

the normalized derivative search profile landmark i will be

$$\mathbf{y}_{s_j} = \frac{ds}{\sum_{k=0}^{n_s-2} |ds_{jk}|} \quad (41)$$

We now examine $\mathbf{y}_{s_j}$ (the normalized derivative search profile) for sub-profiles that match $\bar{\mathbf{y}}_j$ (the mean normalized derivative profile obtained from the training set of images). Denoting the sub-interval of $\mathbf{y}_{s_j}$ centered at the d^{th} pixel of $\mathbf{y}_{s_j}$, by $\mathbf{h}(d)$ we find the value of d that makes the sub-profile $\mathbf{h}(d)$ most similar to $\bar{\mathbf{y}}_j$.

This can be done by defining the following square error function (which decreases as the fit becomes better) and minimizing it with respect to d :

$$f(d) = (\mathbf{h}(d) - \bar{\mathbf{y}}_j)^T (\mathbf{h}(d) - \bar{\mathbf{y}}_j) \quad (42)$$

With the possibility of weighting the error by the inverse of the covariance matrix of the $\mathbf{y}_j$

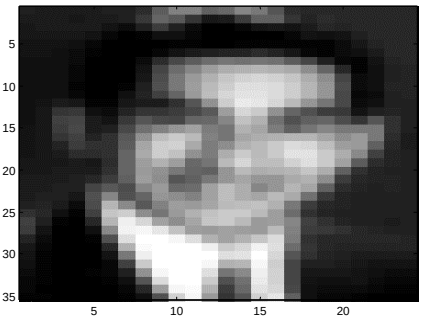
$$f(d) = (\mathbf{h}(d) - \bar{\mathbf{y}}_j)^T \mathbf{C}_{\mathbf{y}_j}^{-1} (\mathbf{h}(d) - \bar{\mathbf{y}}_j) \quad (43)$$

In this way the location of the point to which the landmark i should move is determined. The same procedure is repeated for all the landmark points to obtain the vector of suggested movements $d\mathbf{x}_i$. We may choose to control the suggested movement by, for example, changing a small suggested movement to no movement, and large suggested movements to only half what is suggested.

6. Multi-Resolution Image Search

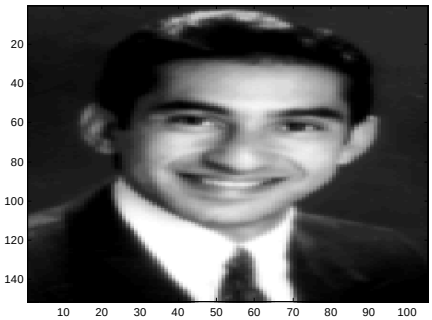
An important issue that affects the image search considerably is the choice of the length of the search profile. In choosing the length of the search profile we are faced with two contradicting requirements; on the one hand, the search profile should be long enough to contain within it the target point (the point that we need the landmark point to move to), on the other, we require the search profile to be short so as, first, to reduce the computations required, and second, if the search profile is long and the target point is close to the current position of the landmark then it will be more probable to move to a far away point and miss the target. This suggests a multi-resolution approach, where at first the search looks for far away points and make large jumps, and as the search homes in on a target structure, it should restrict the search only to only close points.

In order to achieve such multi-resolution search, we generate a pyramid of images. This pyramid of images contains the same image but in different resolutions. At the base of the pyramid (Level 0) we have the original image, one level higher (Level 1) we have a lower resolution version of the image with half the number of pixels along each dimension, and so on, until we reach the highest level of the pyramid (level $L-1$ in an L level pyramid) (see Figure 5).

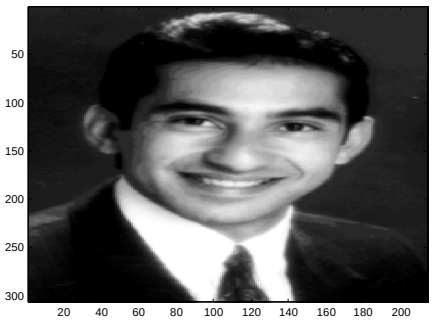


Level: 4

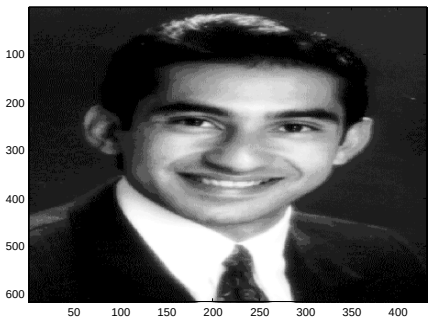
Level: 3



Level: 2



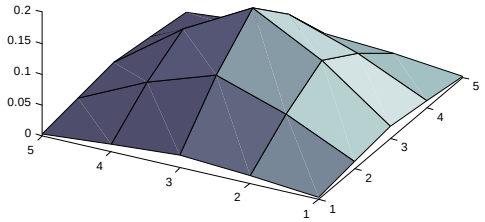
Level: 1



Level: 0

Figure 5. pyramid of images

An example of a possible implementation to obtain the pyramid is to first smooth the image at the lower level with a 5 5 Gaussian mask (which is linearly decomposed into two 1-5-8-5-1 convolutions) (see Figure 6) and then sub-sampling every other pixel to obtain the image at one higher level (Cootes *et al.* [10]).



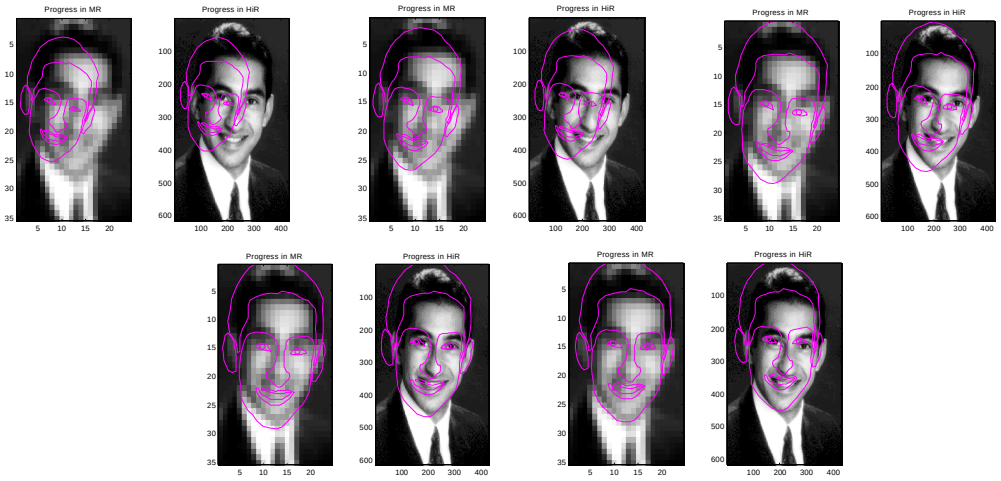
0.125	0.625	1	0.625	0.125
0.625	3.125	5	3.125	0.625
1	5	8	5	1
0.625	3.125	5	3.125	0.625
0.125	0.625	1	0.625	0.125

Figure 6. Gaussian mask

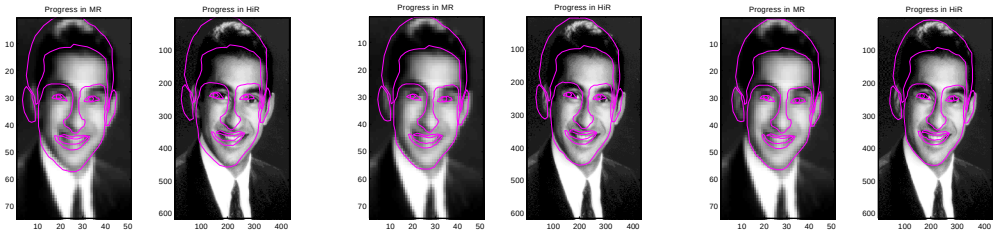
The search will now be performed by first searching at the top level of the pyramid and then initiating the search in one level below using the search output of the previous level, and so on, until the lowest level of the pyramid (the original image) is reached. In order to be able to perform such a search in each level, we should be equipped with information about the gray level profiles in each of these levels, this demands that during the training stage, we should obtain the mean normalized derivative profile for each landmark at all the levels of the pyramid.

The Mean normalized derivative profile for the landmark j at the pyramid level q will be $\bar{\mathbf{y}}_{jq}$ where $1 \le j \le n$ and $0 \le q \le L-1$ since we have L levels in the pyramid and n landmarks in each image. The mean is obtained, as before, by averaging the normalized profiles for a certain landmark along the N images of the training set.

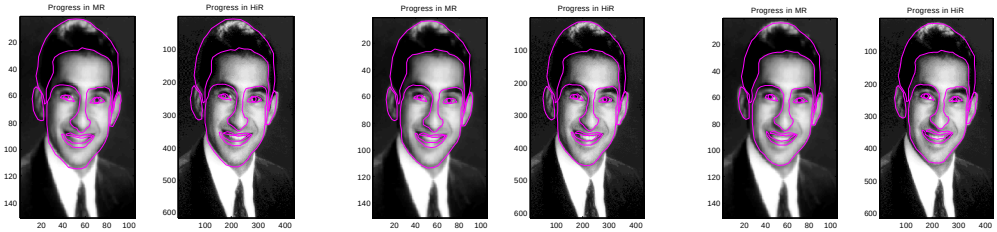
A criterion should be decided upon in order to change the level of search within the pyramid. One example (Cootes *et al.* [10]) is to move to a lower level when a certain percentage of the landmarks did not change considerably, for example when 95% of the landmarks move only within the central 50% of the search profile. A maximum number of iterations can also be devised to prevent from getting stuck at a higher level (See Figure 7 and 8).



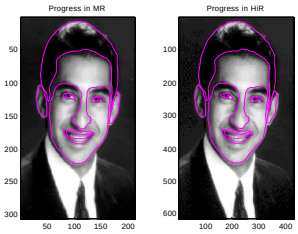
a. search in Level 4



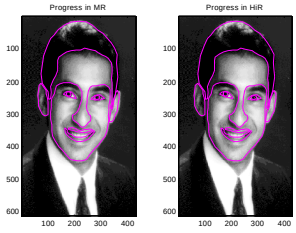
b. search in level 3



c. search in level 2



d. search in level 1



e. search in level 0

Figure 7. The progress of multi resolution image search, (shapes in higher levels are also shown in level 0)

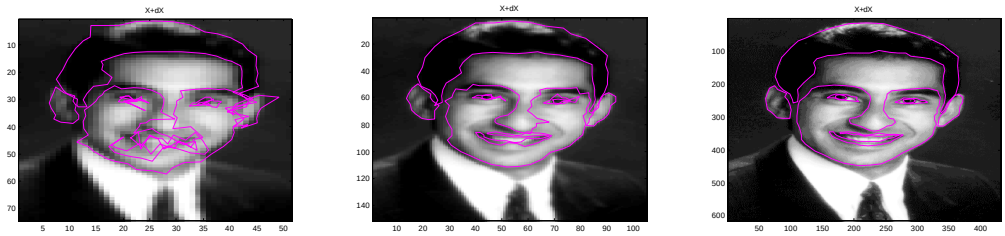


Figure 8. Examples of the desired changes in the shape $\mathbf{x}_i + d\mathbf{x}_i$ *, before forcing it to be an allowable shape. Shown in level 4 (left), level 3 (middle), and level 0 (right)*

7. Shape Recognition and classification

Images of similar objects vary due to two main types of variations, the first is inter-class variations and the second is within-class (intra-class) variations (Lanitis *et al.* [3]). Inter-class variations are the type of changes in the appearance of objects due to the fact that they belong to different classes, for example the variation in the appearance of the faces (due to the individual appearance) of two different persons, assuming that each person represents a class and all images of the same person should be classified to his/hers class. Intra-class variations are changes in appearance of objects although they are in the same class, following the example of human faces, then such type of variation is due to lighting effects, 3D pose or facial expression (Lanitis *et al.* [11]).

The basic idea is to start with a training set of images, obtain shape and gray level information that describe each class, then presented with a new image, search the image for the shape, then obtain the shape and gray level information in this image, and using the training information and discriminant analysis techniques, the new image can be classified.

The shape information is represented by the weights of the principal components, the **b** vector, for a certain point distribution model for the shape of the object in question. The gray level information is divided in to two categories, first local gray level information represented by the normalized derivative profile perpendicular to the landmark points, and the second is shape-free gray level model, obtained by first deforming the shape to the mean shape in such a way that changes in gray levels are minimized. This can be done using thin-plate splines (Bookstein [12]).

8. Conclusions

This report reviewed the shape modeling method known as Active Shape Models. ASMs can be used in image search and image classification. In order to apply ASMs, both shape training and gray-level training must be performed. A prerequisite is to collect a training set of images containing instances of the deformable object to be modeled. Moreover, each object in the training images needs to be labeled by a set of landmarks. Shape training results in a Point Distribution Model (PDM) which is a statistical model of the allowable variations of the shapes of a class of objects. Gray-level training is done to obtain sufficient information about the expected image intensity around the landmark of an object. The gray-level information is then needed in performing image search. In image classification the PDM parameters and certain gray-level information are used as features to assign a new instance of an object to a certain class using discriminant analysis techniques.

ASMs outperforms other deformable models in that it only produces allowable shapes regardless of the level of the noise in an image. The allowable shapes are those whose variations are covered within the training set of images. This suggests that our training set should contain instances of the object to be modeled with only the variability that needs to be modeled. A second advantage is that the algorithms that employ ASMs are general, which means that the same software can be used for a wide variety of applications such as medical an industrial.

Some drawbacks of ASMs need to be mentioned. First, the labeling of the instances found in the training set of images is a tedious task, and when done manually it becomes unacceptable especially with large training sets. Furthermore, the labeling itself should be done as accurately as possible and preferably by an expert. Second, the result of using ASMs in image search is highly dependent on the initial pose of the shape. This may be partially remedied by use of multi-resolution image search, but still for a guaranteed correct convergence ad-hoc methods need to be employed. Third, different parameters need to be specified, an important example is the number of modes of variation to be included in the PDM, other examples such as the specifications on the type of gray-level patches to be studied. In the case of a line profile this maybe the length of such profile.

ASMs seem to be a promising deformable shape modeling technique. Future work to improve ASMs should include attempts to automate or facilitate the labeling of landmarks. Also little work has been done to extend ASMs to cover multi-dimensional data, such as applying ASMs to 3D object modeling and segmentation, or the tracking of time-varying 2D images.

Appendices

Appendix 1. The Weighting Matrix

The weighting matrix is a diagonal matrix of weights. The weights are chosen to give more significance to the landmark points which tend to be more stable. A stable point would move about less with respect to other points in the shape.

To calculate such weights we first calculate the distances between every pair of points in all the shapes. Then we calculate the variance of the distance between every pair of points over all the shapes. Then for a specific point, the sum of the variances of the distances from this point to all others, would measure the instability of the point, we thus take the weight to be the inverse of this summation. Putting this mathematically, let R_{ikl} be the distance between the landmark points k and l in the i^{th} shape. Then let denote the variance of the distance between the landmark points k and l be denoted V_{Rkl}, then the weight for the k^{th} landmark will be

$$w_k = \left(\sum_{l=0}^{n-1} V_{Rkl} \right)^{-1} \text{, where } 0 \le k \le n-1 \qquad \text{(A1. 1)}$$

where n is the number of landmark points.

The weighting matrix will then be the following diagonal matrix.

$$\mathbf{W} = diag(w_1, w_1, w_2, w_{2,\ldots}, w_{n-1}, w_{n-1}) \qquad \text{(A1. 2)}$$

Appendix 2. Aligning Two Shapes

Given two shape-representing vectors $\mathbf{x}_1$ and $\mathbf{x}_2$,

$$\mathbf{x}_1 = \left[x_{10}, y_{10}, x_{11}, y_{11}, \ldots, x_{1k}, y_{1k}, \ldots x_{1(n-1)}, y_{1(n-1)}\right]^T \quad (\text{A2. 1})$$

$$\mathbf{x}_2 = \left[x_{20}, y_{20}, x_{21}, y_{21}, \ldots, x_{2k}, y_{2k}, \ldots x_{2(n-1)}, y_{2(n-1)}\right]^T \quad (\text{A2. 2})$$

where each vector contains the (x, y) coordinates of n landmarks, we would like to find

- the rotation angle: q,
- the scaling factor: s,
- and the value of the translation in both dimensions: (t_x, t_y)

that will align $\mathbf{x}_2$ to $\mathbf{x}_1$, by mapping $\mathbf{x}_2$ to $\hat{\mathbf{x}}_2$,

$$\hat{\mathbf{x}}_2 = \left[\hat{x}_{20}, \hat{y}_{20}, \hat{x}_{21}, \hat{y}_{21}, \ldots, \hat{x}_{2k}, \hat{y}_{2k}, \ldots \hat{x}_{2(n-1)}, \hat{y}_{2(n-1)}\right]^T \quad (\text{A2. 3})$$

Using

$$\hat{\mathbf{x}}_2 = M(s,q)\left[\mathbf{x}_2\right] + \mathbf{t} \quad (\text{A2. 4})$$

where

$M(s,q)\left[\mathbf{x}_2\right]$ is a rotated then scaled version of $\mathbf{x}_2$ (by q and s respectively)

and

$\mathbf{t} = \left[t_x \quad t_y \quad t_x \quad t_y \quad \ldots \quad t_x \quad t_y\right]^T$ is the translation vector and is of length $2n$.

the weighted distance between the two points in the $2n$-D space representing the two shapes $\mathbf{x}_1$ and $\hat{\mathbf{x}}_2$, is given by:

$$d_{1\hat{2}}^2 = w_{0x}^2\left(\hat{x}_{20} - x_{10}\right)^2 + w_{0y}^2\left(\hat{y}_{20} - y_{10}\right)^2 + w_{1x}^2\left(\hat{x}_{21} - x_{11}\right)^2 + w_{1y}^2\left(\hat{y}_{21} - y_{11}\right)^2 +$$
$$\ldots + w_{(n-1)x}^2\left(\hat{x}_{2(n-1)} - x_{1(n-1)}\right)^2 + w_{(n-1)y}^2\left(\hat{y}_{2(n-1)} - y_{1(n-1)}\right)^2$$

or in matrix form

$$d_{1\hat{2}}^2 = (\hat{\mathbf{x}}_2 - \mathbf{x}_1)^T \mathbf{W}^T \mathbf{W} (\hat{\mathbf{x}}_2 - \mathbf{x}_1) \quad (\text{A2. 5})$$

where

$\mathbf{W} = diag(w_{ox}, w_{oy}, w_{1x}, w_{1y}, \ldots, w_{(n-1)x}, w_{(n-1)y})$

substituting $\hat{\mathbf{x}}_2 = M(s,q)\left[\mathbf{x}_2\right] + \mathbf{t}$ we have

$$d_{1\hat{2}}^2 = (M(s,q)\left[\mathbf{x}_2\right] + \mathbf{t} - \mathbf{x}_1)^T \mathbf{W}^T \mathbf{W} (M(s,q)\left[\mathbf{x}_2\right] + \mathbf{t} - \mathbf{x}_1) \quad (\text{A2. 6})$$

For the rotation, scaling and translation of a single coordinate (x_{2k}, y_{2k}) we have

$$\begin{bmatrix}\hat{x}_{2k} \\ \hat{y}_{2k}\end{bmatrix} = M(s,q)\begin{bmatrix}x_{2k} \\ y_{2k}\end{bmatrix} + \begin{bmatrix}t_x \\ t_y\end{bmatrix} = \begin{bmatrix}s\cos(q) & -s\sin(q) \\ s\sin(q) & s\cos(q)\end{bmatrix}\cdot\begin{bmatrix}x_{2k} \\ y_{2k}\end{bmatrix} + \begin{bmatrix}t_x \\ t_y\end{bmatrix} \quad (\text{A2. 7})$$

and denoting $a_x = s\cos(q)$ and $a_y = s\sin(q)$ we have

$$\begin{bmatrix}\hat{x}_{2k} \\ \hat{y}_{2k}\end{bmatrix} = \begin{bmatrix}a_x & -a_y \\ a_y & a_x\end{bmatrix}\cdot\begin{bmatrix}x_{2k} \\ y_{2k}\end{bmatrix} + \begin{bmatrix}t_x \\ t_y\end{bmatrix} \quad (\text{A2. 8})$$

Now, we can rewrite $\hat{\mathbf{x}}_2$ as

$$\hat{\mathbf{x}}_2 = \begin{bmatrix} \begin{bmatrix} a_x & -a_y \\ a_y & a_x \end{bmatrix} \cdot \begin{bmatrix} x_{20} \\ y_{20} \end{bmatrix} + \begin{bmatrix} t_x \\ t_y \end{bmatrix} \\ \begin{bmatrix} a_x & -a_y \\ a_y & a_x \end{bmatrix} \cdot \begin{bmatrix} x_{21} \\ y_{21} \end{bmatrix} + \begin{bmatrix} t_x \\ t_y \end{bmatrix} \\ \vdots \\ \begin{bmatrix} a_x & -a_y \\ a_y & a_x \end{bmatrix} \cdot \begin{bmatrix} x_{2k} \\ y_{2k} \end{bmatrix} + \begin{bmatrix} t_x \\ t_y \end{bmatrix} \\ \vdots \\ \begin{bmatrix} a_x & -a_y \\ a_y & a_x \end{bmatrix} \cdot \begin{bmatrix} x_{2(n-1)} \\ y_{2(n-1)} \end{bmatrix} + \begin{bmatrix} t_x \\ t_y \end{bmatrix} \end{bmatrix} = \begin{bmatrix} a_x x_{20} - a_y y_{20} + t_x \\ a_y x_{20} + a_x y_{20} + t_y \\ a_x x_{21} - a_y y_{21} + t_x \\ a_y x_{21} + a_x y_{21} + t_y \\ \vdots \\ a_x x_{2k} - a_y y_{2k} + t_x \\ a_y x_{2k} + a_x y_{2k} + t_y \\ \vdots \\ a_x x_{2(n-1)} - a_y y_{(2n-1)} + t_x \\ a_y x_{2(n-1)} + a_x y_{2(n-1)} + t_y \end{bmatrix} = \begin{bmatrix} x_{20} & -y_{20} & 1 & 0 \\ y_{20} & x_{20} & 0 & 1 \\ x_{21} & -y_{21} & 1 & 0 \\ y_{21} & x_{21} & 0 & 1 \\ \vdots & \vdots & \vdots & \vdots \\ x_{2k} & -y_{2k} & 1 & 0 \\ y_{2k} & x_{2k} & 0 & 1 \\ \vdots & \vdots & \vdots & \vdots \\ x_{2(n-1)} & -y_{2(n-1)} & 1 & 0 \\ y_{2(n-1)} & x_{2(n-1)} & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} a_x \\ a_y \\ t_x \\ t_y \end{bmatrix}$$

which can be written as

$$\hat{\mathbf{x}}_2 = \mathbf{A}\mathbf{z} \qquad (\text{A2. 9})$$

where

$$\mathbf{A} = \begin{bmatrix} x_{20} & -y_{20} & 1 & 0 \\ y_{20} & x_{20} & 0 & 1 \\ x_{21} & -y_{21} & 1 & 0 \\ y_{21} & x_{21} & 0 & 1 \\ \vdots & \vdots & \vdots & \vdots \\ x_{2k} & -y_{2k} & 1 & 0 \\ y_{2k} & x_{2k} & 0 & 1 \\ \vdots & \vdots & \vdots & \vdots \\ x_{2(n-1)} & -y_{2(n-1)} & 1 & 0 \\ y_{2(n-1)} & x_{2(n-1)} & 0 & 1 \end{bmatrix}$$

and

$$\mathbf{z} = \left[a_x, a_y, t_x, t_y\right]^T$$

Rewriting

$$d_{1\hat{2}}^2 = (M(s,q)\left[\mathbf{x}_2\right] + \mathbf{t} - \mathbf{x}_1)^T \mathbf{W}^T \mathbf{W} (M(s,q)\left[\mathbf{x}_2\right] + \mathbf{t} - \mathbf{x}_1)$$

as

$$d_{1\hat{2}}^2 = (\mathbf{A}\mathbf{z} - \mathbf{x}_1)^T \mathbf{W}^T \mathbf{W} (\mathbf{A}\mathbf{z} - \mathbf{x}_1) \qquad (\text{A2. 10})$$

we can now solve for $\mathbf{z}$ that minimizes $d_{1\hat{2}}^2$, which is the least squares solution to $\mathbf{W}\mathbf{A}\mathbf{z} = \mathbf{W}\mathbf{x}_1$, (Strang [8, page 204]),

$$\mathbf{z} = ((\mathbf{W}\mathbf{A})^T (\mathbf{W}\mathbf{A}))^{-1} (\mathbf{W}\mathbf{A})^T \mathbf{W}\mathbf{x}_1) = (\mathbf{A}^T \mathbf{W}^T \mathbf{W}\mathbf{A})^{-1} \mathbf{A}^T \mathbf{W}^T \mathbf{W}\mathbf{x}_1 \qquad (\text{A2. 11})$$

Once $\mathbf{z} = \left[a_x, a_y, t_x, t_y\right]^T$ is known, s and q can be found since

$$\frac{a_y}{a_x} = \frac{s\sin(q)}{s\cos(q)} = \tan(q) \Rightarrow q = \arctan(\frac{a_y}{a_x}) \quad \text{(A2. 12)}$$

$$a_x = s\cos(q) \Rightarrow s = \frac{a_x}{\cos(\arctan(\frac{a_y}{a_x}))} \quad \text{(A2. 13)}$$

A different formulation for solving the problem of aligning two shapes is presented by Cootes *et al.* [1].

Appendix 3. Principal Component Analysis

We assume that we have an observation matrix $\mathbf{X}$ of size $m \times n$, where m is the number of observations and n is the number of variables in each observation, *i.e.* each row of the observation matrix is an observation and each column is a variable (in our previous discussions we had N observations and $2n$ variables in the training set of images).

One way of obtaining the principal components is using eigenvalue decomposition of the covariance matrix of the observation matrix done by Cootes *et al.* [1].

The covariance matrix can be obtained by

$$\mathbf{S} = \frac{1}{m}\sum_{i=1}^{m}(\mathbf{x}_i - \bar{\mathbf{x}})^T(\mathbf{x}_i - \bar{\mathbf{x}}) \qquad (A3.\ 1)$$

where

$$\bar{\mathbf{x}} = \frac{1}{m}\sum_{i=1}^{m}\mathbf{x}_i \qquad (A3.\ 2)$$

The unit eigenvectors $\mathbf{p}_k$; $1 \le k \le n$, of $\mathbf{S}$ are such that

$$\mathbf{S}\mathbf{p}_k = l_k\mathbf{p}_k \qquad (A3.\ 3)$$

where
l_k is the k^{th} eigenvalue of $\mathbf{S}$, $l_k \ge l_{k+1}$, and $\mathbf{p}_k{}^T\mathbf{p}_k = 1$

Assuming that the observations form a hyper ellipsoid in n-Dimensions, then the eigenvectors of the covariance matrix corresponding to the largest eigenvalues describe the longest axes of the ellipsoid, and thus the most significant modes of variations in the variables used to derive the covariance matrix, and the variance explained by each eigenvector is equal to the corresponding eigenvalue. We can approximate the n-Dimensional ellipsoid by a t-Dimensional ellipsoid, and thus take the first t eigenvectors as the first t principal components.

Another method to obtain the principal components is by performing a Singular Value Decomposition (SVD) of the observation matrix centered around its mean [13]. First we center the observation matrix $\mathbf{X}$ around its mean and obtain

$$\mathbf{X}_c = \mathbf{X} - \bar{\mathbf{x}} \cdot \mathbf{1}^T \qquad (A3.\ 4)$$

where $\mathbf{1}^T = \begin{bmatrix}1 & 1 & \dots & 1\end{bmatrix}$ and is of length n.

then we perform an 'economy size' SVD of $\frac{1}{\sqrt{m-1}}\mathbf{X}_c$ and obtain $\mathbf{U},\mathbf{S},\mathbf{V}$

$$[\mathbf{U},\mathbf{S},\mathbf{V}] = svd\left(\frac{1}{\sqrt{m-1}}\mathbf{X}_c, 0\right) \qquad (A3.\ 5)$$

such that

$$\mathbf{X}_c = \mathbf{U} \cdot \mathbf{S} \cdot \mathbf{V}^T \qquad (A3.\ 6)$$

where the principal components are the columns of the $\mathbf{V}$ matrix, and the eigenvalues of the covariance are contained in $\mathbf{S}$.

References

[1] Cootes T, Taylor C, Cooper D, Graham J. Active Shape Models - Their Training and Application. Computer Vision and Image Understanding, January 1995, Vol. 61, No. 1, pp. 38-59.

[2] M. Kass, A. Witkin, D. Terzopoulos, Snakes: Active Contour Models, In Proceedings, First International Conference On Computer Vision, pp. 259-268, IEEE Comput. Soc. Press, 1987.

[3] Lanitis A, Taylor C, Cootes T. Automatic Interpretation and Coding of Face Images Using Flexible Models. IEEE Transactions on Pattern Analysis and Machine Intelligence, July 1997, Vol. 19, No. 7, pp. 743-756.

[4] The ORL Database of Faces, http://www.cam-orl.co.uk/facedatabase.html

[5] Mardia K, Kent J, Bibby J. Multivariate Analysis, Academic Press, 1995, pp. 213-254

[6] Bradley J. MATLAB Statistics Toolbox - User's Guide, The MathWorks Inc., 1997, Tutorial pp. 77-87.

[7] Mardia K, Kent J, Bibby J. Multivariate Analysis, Academic Press, 1995, pp. 213-254

[8] Strang G. Linear Algebra And Its Applications, Saunders 1988.

[9] Cootes T, Taylor C, Hill A, Halsam J. The Use of Active Shape Models for Locating Structures in Medical Images. Proceedings of th 13th International Conference on Information Processing in Medical Imaging, (Eds. H.H.Barrett, A.F.Gmitro) Springer-Verlag, 1993, pp. 33-47.

[10] Cootes T, Taylor C, Lanitis A. Active Shape Models: Evaluation of a Multi-Resolution Method for Improving Image Search. Proceedings of the British Machine Vision Conference, 1994, pp.327-336.

[11] Lanitis A, Taylor C, Cootes T. Recognising Human Faces Using Shape and Grey-Level Information. Third International Conference on Automation, Robotics and Computer Vision, 1994. ??-??

[12] Bookstein F. Principal Warps: Thin-Plate Splines and the Decomposition of Deformations, IEEE Transactions on Pattern Analysis and Machine Intelligence, VOL 11, No. 6, June 1989, pp. 567-585.

[13] MATLAB version 5.1, and the 'rudimentary statistics toolbox' more specifically the functions 'princomp(…)' and 'svd(…)' MATLAB's reference: J. Edward Jackson, A User's Guide to Principal Components, John Wiley & Sons, Inc. 1991 pp. 1-25, B. Jones 3-17-94.