An Introduction to Digital Video Data Compression in Java

Fore June

Chapter 16 Active Shape Models (ASMs)

16.1 Introduction

Active shape models (ASMs) are statistical models for image processing and recognition, developed by Tim Cootes and Chris Taylor in the 1990s. ASM closely relates to the active appearance model (AAM), which is also known as a **smart snake** method. The formal name for **snake** model is **active contour model**.

A real-world image often consists of complex objects. Two image objects representing the same real-world object may vary in appearance and shape from one image to another. It is an inherent difficult task to recognize the existence of certain structures in an image. There are a lot of studies and methods of locating known objects in images. The method of using rigid models to represent image objects is well established. However, in many practical situations rigid models are not appropriate because objects of the same class are not identical. For example, in medical applications, the shape of organs may vary significantly through time and between individuals. In many industrial image processing applications, the images may involve assemblies with moving parts, and/or components with varying appearance. In such cases, we have to use flexible models, or deformable templates to allow for some degree of variability in the shape of the imaged objects. One may use trigonometric functions such as sin and cos to describe shapes. By varying the parameters and the number of terms used in a trigonometric series, one can generate different shapes. However, such methods are not suitable for describing general shapes. For example, using a finite number of terms, we can define a square corner only approximately. There is no clear relationship between variations in shape and variations in the parameters of the trigonometric expansion.

Utilizing models that cope with the variability, ASMs are able to remedy the defects of rigid models and are able to identify complex objects and special features of an image, and find examples of the structures that they represent.

An active shape model makes use of a set of annotated images of typical examples to build a statistical model of appearance. It requires one to first decide upon a suitable set of points (landmarks) to describe the shape of the target; the landmarks should be found reliably on each training image. The set of points representing each object or image structure may represent boundaries, internal features, or even external structures, such as the center of a concave boundary of a region. In the method, one has to manually place the points in the same way on each of a training set of examples of the object. The points that mark significant positions on an image object are usually referred to as *landmarks*. Each landmark point represents a distinguishable point on every example image. For example, when we build a model of the appearance of an eye of a human face image, we could choose the corners of the eye as landmarks as they are easy to identify and mark in an image. Such a requirement constrains the application of the method as the object shapes involve cannot change abruptly from image to image. Therefore, the method is not appropriate for highly amorphous objects such as some types of cells or simple organisms.

In our application, we use a simple OpenGL program to display an image, and use the mouse to click on the desired points, which are captured by the program and saved in a file. The program minimizes the variance in distance between equivalent points by automatically align the sets of points. The principal component analysis (PCA) technique discussed in the previous chapter is used to reduce data redundancy. By analyzing the point distribution, a model is derived to give the average positions of the points and a number of parameters to control the main modes of variation contained in the training sets.

Points at clear corners of object boundaries or 'T' junctions are good choices for landmarks. In practice, to make a good description of the shape of a target object, one needs to choose a large number of landmark points. Moreover, one should augment a landmark list with points along boundaries and these points should be placed equally spaced between well defined landmark points. However, in our examples here, landmarks are created very briefly without the augmented features; the main purpose of the examples is to illustrate some basic techniques and principles of ASM. Figure 16-1 below shows a face image annotated with landmarks.

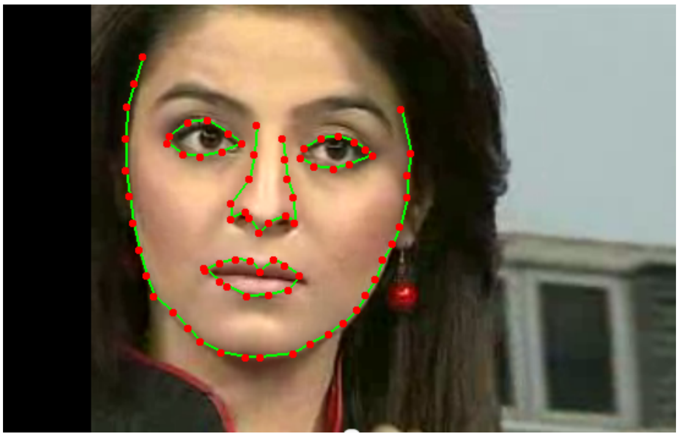
Figure 16-1 A Face Image Annotated with Landmarks

16.2 Statistical Models

We consider two dimensional images. We label significant points referred to as *landmarks* in images of interest in order to examine and measure shape changes which could be correlated with other factors. The landmarks, which are representative points may capture shape constraints and will be used to build models. We can then use the models to construct plausible new shape examples for use in image interpretation.

We define a point p_i by its x-y coordinates:

$$p_i = \begin{pmatrix} x_i \\ y_i \end{pmatrix} \tag{16.1}$$

We define a shape S by a set of n points:

$$S = \{p_1, p_2, \cdots, p_n\} \tag{16.2}$$

Each point in S is usually referred to as a *landmark*, which "marks" a significant position of an image object.

We can form a linear affine combination of points by requiring the sum of the combining coefficients to be equal to 1. That is.

$$p = \alpha_1 p_1 + \alpha_2 p_2 + \cdots + \alpha_n p_n = \begin{pmatrix} \alpha_1 x_1 + \alpha_2 x_2 + \cdots + \alpha_n x_n \\ \alpha_1 y_1 + \alpha_2 y_2 + \cdots + \alpha_n y_n \end{pmatrix} \tag{16.3}$$

is a legitimate point if $\alpha_1 + \alpha_2 + \cdots + \alpha_n = 1$.

Suppose we have N aligned shapes, and each shape S_k is defined by an equation of (16.2); we can calculate the mean shape $\overline{S}$ by

$$\overline{S} = \frac{1}{N}\sum_{k=1}^{N} S_k \tag{16.4}$$

In (16.4), each combining coefficient is $\alpha_k = \frac{1}{N}$. Equation (16.4) means that for each point in the shape, we take the average of N points from N shapes. For example, the i-th point, $\overline{p}_i$ of $\overline{S}$ is given by

$$\overline{p}_i = \frac{1}{N}\sum_{k=1}^{N} p_i^k \tag{16.5}$$

where p_i^k is the i-th point of the k-th shape, S_k.

We now consider a shape S_k as one super-point with dimension $2n$. That is,

$$S_k = \left(x_1^k, \cdot\cdot, x_n^k, y_1^k, \cdot\cdot, y_n^k\right) \tag{16.6}$$

In the forthcoming discussions, when there is no confusion, we may simply refer to a super-point as a point. Therefore, a training set of N shapes is composed of N points in $2n$ dimensions. We can apply a principal component analysis (PCA) to these N points in the usual manner discussed in the previous chapter. Each axis indicates a way that the landmark points tend to move together as the shape changes.

For each super-point (shape) S_k in the training set we can calculate its deviation, d_k, from the mean, $\overline{S}$:

$$d_k = S_k - \overline{S} = \left(x_1^k - \overline{x}_1, \cdot\cdot, x_n^k - \overline{x}_n, y_1^k - \overline{y}_1, \cdot\cdot, y_n^k - \overline{y}_n\right) \tag{16.7}$$

Each d_k is a $1 \times 2n$ row-vector. The deviation matrix D is an $N \times 2n$ matrix given by

$$D = \begin{pmatrix} d_1 \\ d_2 \\ \cdot \\ \cdot \\ d_N \end{pmatrix} \tag{16.8}$$

Though d_i is a row-vector, we can denote d_{ij} as the ij-th element of matrix D without confusion. Note that the transpose of D, denoted by D^T, is a $2n \times N$ matrix.

We can then calculate the $2n \times 2n$ covariance matrix C using (15.36), where the denominator would be $N - 1$, the number of shapes minus 1. However, to be consistent with the calculations used by other authors in the field, we use N in the denominator:

$$C = \frac{1}{N} D^T D \tag{16.9}$$

where D^T is the transpose of D and the ij-th element, c_{ij}, of C is given by

$$c_{ij} = \frac{1}{N}\sum_{k=1}^{N} d_{ki} d_{kj} \tag{16.10}$$

Matrix C is symmetric and is $2n \times 2n$; it has $2n$ eigenvectors. Following the conventions we have used in Chapter 15, each eigenvector $\mathbf{e_i}$ is a $1 \times 2n$ row vector. The projection matrix P is given by

$$P = \begin{pmatrix} \mathbf{e_1} \\ \cdot \\ \cdot \\ \mathbf{e_{2n}} \end{pmatrix} \tag{16.11}$$

which is a $2n \times 2n$ square matrix. We define a training set $\mathbf{X_k}$ as the transpose of the shape S_k. So $\mathbf{X_k}$ is a $2n \times 1$ column vector:

$$\mathbf{X_k} = S_k^T = \begin{pmatrix} x_1^k \\ \cdot \\ \cdot \\ x_n^k \\ y_1^k \\ \cdot \\ \cdot \\ y_n^k \end{pmatrix} \tag{16.12}$$

and the mean of the training sets is $\overline{\mathbf{X}} = \overline{S}^T$. We can name the transpose of the deviation matrix D as A, which is a $2n \times N$ matrix:

$$A = D^T = \left(\mathbf{X_1} - \overline{\mathbf{X}}, \cdot\cdot, \mathbf{X_N} - \overline{\mathbf{X}}\right) \tag{16.13}$$

As presented in (15.56) of Chapter 15, the projection of A onto the new basis (eigenvectors) is given by:

$$\begin{aligned} B = PA \quad &= \begin{pmatrix} \mathbf{e_1} \\ \cdot \\ \cdot \\ \mathbf{e_{2n}} \end{pmatrix} \left(\mathbf{X_1} - \overline{\mathbf{X}}, \cdot\cdot, \mathbf{X_N} - \overline{\mathbf{X}}\right) \\ &= \begin{pmatrix} \mathbf{e_1} \cdot (\mathbf{X_1} - \overline{\mathbf{X}}), & \cdot\cdot, & \mathbf{e_1} \cdot (\mathbf{X_N} - \overline{\mathbf{X}}) \\ \cdot & \cdot\cdot & \cdot \\ \cdot & \cdot\cdot & \cdot \\ \mathbf{e_{2n}} \cdot (\mathbf{X_1} - \overline{\mathbf{X}}), & \cdot\cdot, & \mathbf{e_{2n}} \cdot (\mathbf{X_N} - \overline{\mathbf{X}}) \end{pmatrix} \end{aligned} \tag{16.14}$$

which is a $2n \times N$ matrix. (Note that each $\mathbf{e_i}$ is $1 \times 2n$, and each $(\mathbf{X_i} - \overline{\mathbf{X}})$ is $2n \times 1$. So $e_i \cdot (X_i - \overline{X})$ is 1×1, which is a scalar.) If we apply PCA to the data and only retain the first t principal eigenvectors ($t < 2n$), the projection matrix P becomes F, and (16.14) is reduced to:

$$B = FA \quad = \begin{pmatrix} \mathbf{e_1} \cdot (\mathbf{X_1} - \overline{\mathbf{X}}), & \cdot\cdot, & \mathbf{e_1} \cdot (\mathbf{X_N} - \overline{\mathbf{X}}) \\ \cdot & \cdot\cdot & \cdot \\ \cdot & \cdot\cdot & \cdot \\ \mathbf{e_t} \cdot (\mathbf{X_1} - \overline{\mathbf{X}}), & \cdot\cdot, & \mathbf{e_t} \cdot (\mathbf{X_N} - \overline{\mathbf{X}}) \end{pmatrix} \tag{16.15}$$

Here, F, A and B are $t \times 2n$, $2n \times N$, and $t \times N$ respectively. If we denote the k-th column vector of B as $\mathbf{b_k}$, then

$$\mathbf{b_k} = \begin{pmatrix} \mathbf{e_1} \cdot (\mathbf{X_k} - \overline{\mathbf{X}}) \\ \cdot \\ \cdot \\ \mathbf{e_t} \cdot (\mathbf{X_k} - \overline{\mathbf{X}}) \end{pmatrix} = F(\mathbf{X_k} - \overline{\mathbf{X}}) \tag{16.16}$$

The original shape $\mathbf{X_k}$ is approximated by

$$\mathbf{X_k} \approx \overline{\mathbf{X}} + F^T \mathbf{b_k} \tag{16.17}$$

Conversely, the t-dimensional vector $\mathbf{b_k}$ can be expressed as

$$\mathbf{b_k} \approx F(\mathbf{X_k} - \overline{\mathbf{X}}) \tag{16.18}$$

If $F = P$ (i.e. $t = 2n$), then $F^T = P^T = P^{-1}$ and the original data can be recovered exactly.

We can generalize $\mathbf{b_k}$ to a t-dimensional vector $\mathbf{b}$ which defines a set of parameters of a deformable model. A specific shape $\mathbf{X}$ (a t-dimensional column vector) can be obtained by varying the elements of $\mathbf{b}$; the shape is calculated by

$$\mathbf{X} = \overline{\mathbf{X}} + F^T \mathbf{b} \tag{16.19}$$

We denote the i-th element of $\mathbf{b}$ as b_i. That is,

$$b_i = \mathbf{e_i} \cdot (\mathbf{X} - \overline{\mathbf{X}})$$

Note that b_i is just one element of the column vector $\mathbf{b}$; do not confuse this with $\mathbf{b_k}$, which is simply a shape in the training set, a special $\mathbf{b}$. Suppose the variance of b_i across the training set is δ_i. We can ensure that the shape generated is similar to those of the original set by limiting b_i to vary within the limits of 3 standard deviations, $\pm 3\sqrt{\delta_i}$. People usually call the model variation corresponding to b_i as the i-th *mode* of the model. The feature vector matrix F, consisting of principal eigenvectors of the covariance matrix C, defines a rotated coordinate with each of its axis aligned with a cloud of the original shape vectors. The vector $\mathbf{b}$ defines points in this rotated coordinate system.

16.3 PCA with Fewer Samples than Dimensions

The above PCA technique works well when the number of training shapes N is larger than the vector dimension ($2n$). However, if the number of shapes used is significantly smaller than the vector dimension, the method becomes inefficient as many of the eigenvector components are 0.

Suppose we wish to apply a PCA to N points (shapes) each with $2n$ components, where $N < 2n$. The covariance matrix C, given by (16.9), is $2n \times 2n$, which may be very large. However, we can compute the eigenvalues and eigenvectors from a smaller matrix with order $N \times N$, derived from the data. Operations on a smaller matrix could save a significant amount of computing cost because such operations often go as the cube of the size of the matrix.

We start by subtracting each data vector X_k of (16.12) from the mean $\overline{\mathbf{X}}$ and put them in the transpose of the deviation matrix D^T, which is the same matrix shown in (16.13):

$$D^T = \left(\mathbf{X_1} - \overline{\mathbf{X}}, \cdot\cdot, \mathbf{X_N} - \overline{\mathbf{X}}\right) \tag{16.20}$$

Matrix D^T is $2n \times N$ and D is $N \times 2n$.. The covariant matrix C is given by

$$C = \frac{1}{N} D^T D \tag{16.21}$$

which is $2n \times 2n$. Normally, it has $2n$ eigenvectors and eigenvalues. However, if $N < 2n$, many of the eigenvalues are zero.

Suppose we calculate a matrix T from

$$T = \frac{1}{N} D D^T \tag{16.22}$$

which is $N \times N$ and is much smaller than C. Let $\{\mathbf{e_1}, \cdot, \mathbf{e_i}, \cdot, \mathbf{e_N}\}$ be the set of N eigenvectors of T with corresponding eigenvalues $\{\lambda_1, \cdot, \lambda_i, \cdot, \lambda_N\}$. Each eigenvector $\mathbf{e_i}$ is a $1 \times N$ row vector. The product $\mathbf{e'_i} = \mathbf{e_i} D$ is a $1 \times 2n$ row vector. One can show that $\mathbf{e'_i}$ is an eigenvector of C with corresponding eigenvalue λ_i. There are N such eigenvectors; all the remaining $2n - N$ eigenvectors of C have zero eigenvalues. The vector $\mathbf{e'_i}$ may not be of unit length; we may need to normalize it to make comparisons.

The feature vector matrix is given by

$$F = \begin{pmatrix} \mathbf{e'_1} \\ \cdot \\ \cdot \\ \mathbf{e'_t} \end{pmatrix} = \begin{pmatrix} \mathbf{e_1} D \\ \cdot \\ \cdot \\ \mathbf{e_t D} \end{pmatrix} \tag{16.23}$$

which is $t \times 2n$.

16.3 Shape Model Example

Figure 16-2 below shows shapes from a training set of 6 landmarked faces. Each image is annotated with 81 landmarks and is displayed in a window of 500 pixels $\times$ 500 pixels.

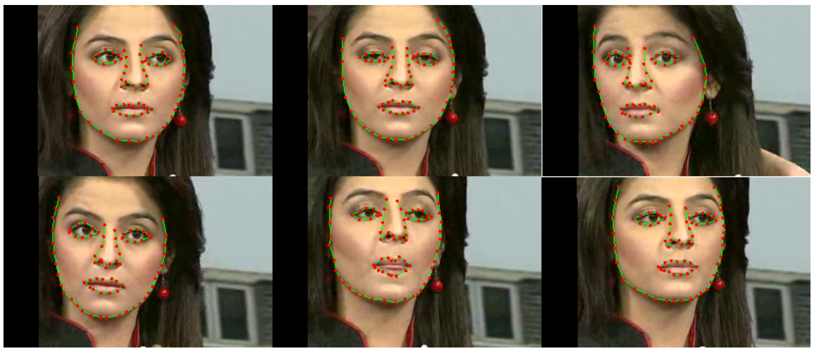
Figure 16-2 Shapes from a Training Set of Faces

Figure 16-3 shows the outlines of the 6 shapes drawn from the landmarks with the centroid of each shape located at the same origin of the drawing coordinate system. The thick black outline in the figure is the average ($\overline{\mathbf{X}}$) of the 6 shapes.

In this example, the number of samples N is 6 and the number of dimensions $2n$ is $2 \times 81 = 162$. So we shall use matrix T given by equation (16.22) to determine the 6 eigenvectors and eigenvalues. As one can see from Figure 16-2, the landmarks can be separated into 5 groups: face, mouth, nose, left eye, and right eye. So we should expect that the data would cluster around 5 axes (eigenvectors) and thus one of the eigenvalues should be very small, close to 0.

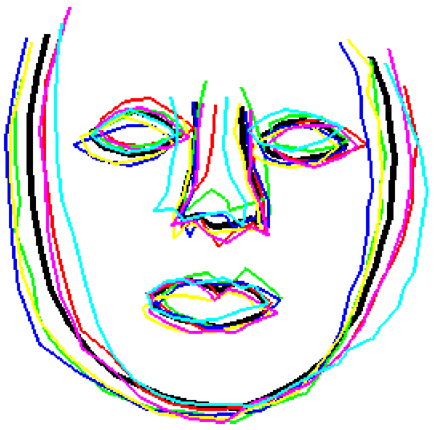

Figure 16-3 Outlines of Six Shapes and Their Mean

We use the Jacobi Method discussed in the previous chapter to find eigenvectors and eigenvalues. The six eigenvalues, arranged from large to small, are found to be:

$$\begin{array}{lll}\lambda_1 = 13846.922, & \lambda_2 = 3538.559, & \lambda_3 = 2090.0279,\\ \lambda_4 = 968.843, & \lambda_5 = 618.593, & \lambda_6 = 0.000\end{array}$$

As an example, the values of the normalized eigenvector $\mathbf{e_1}'$ for λ_1 are:

```
  0.089  0.086  0.093  0.083  0.078   0.077  0.046 -0.011 -0.075 -0.144
 -0.176 -0.197 -0.208 -0.206 -0.186  -0.164 -0.117 -0.097 -0.069 -0.043
 -0.016 -0.008  0.018  0.032  0.058   0.083  0.104  0.111  0.110  0.127
  0.004  0.009  0.005  0.002 -0.014  -0.022 -0.030  0.018  0.014 -0.004
 -0.011 -0.017 -0.007 -0.018  0.003   0.004  0.054  0.055  0.051  0.053
  0.057  0.010 -0.001  0.014  0.027   0.054  0.037  0.030  0.020  0.037
  0.034  0.074  0.083  0.096  0.066   0.031  0.045  0.026  0.021 -0.001
  0.018  0.017  0.019 -0.012 -0.014  -0.012  0.008  0.012  0.006  0.020
  0.014  0.059  0.070  0.099  0.120   0.131  0.150  0.193  0.180  0.169
  0.162  0.129  0.096  0.040  0.007  -0.036 -0.069 -0.073 -0.101 -0.128
 -0.151 -0.159 -0.178 -0.188 -0.171  -0.183 -0.178 -0.146 -0.123 -0.115
 -0.063  0.030  0.024  0.024  0.017  -0.004  0.023  0.026  0.027  0.007
  0.005 -0.001  0.012  0.017  0.038   0.022  0.030  0.042  0.062  0.045
  0.024  0.015  0.015  0.028  0.047   0.054  0.042  0.018  0.017  0.016
  0.026  0.022  0.038  0.028  0.008   0.002  0.003  0.024  0.007  0.006
  0.008  0.009  0.048 -0.003 -0.025  -0.003  0.009 -0.035 -0.014 -0.016
 -0.008 -0.012
```

Suppose we choose the first 3 eigenvectors $\mathbf{e_1}, \mathbf{e_2}$, and $\mathbf{e_3}$ to form the feature vector F:

$$F = \begin{pmatrix} \mathbf{e}'_1 \\ \mathbf{e}'_2 \\ \mathbf{e}'_3 \end{pmatrix} \tag{16.24}$$

So $t = 3$ and F is a 3×162 matrix. A shape projected onto these axes (eigenvectors) is given by

$$\mathbf{b} = F(\mathbf{X} - \overline{\mathbf{X}}) = \begin{pmatrix} \mathbf{e}'_1 \cdot (\mathbf{X} - \overline{\mathbf{X}}) \\ \mathbf{e}'_2 \cdot (\mathbf{X} - \overline{\mathbf{X}}) \\ \mathbf{e}'_3 \cdot (\mathbf{X} - \overline{\mathbf{X}}) \end{pmatrix} \tag{16.25}$$

which is a 3×1 column matrix.

The original shape can be 'recovered' by (16.9) which is

$$\mathbf{X} = \overline{\mathbf{X}} + F^T \mathbf{b}$$

where F^T is the the transpose of F and is a 162×3 matrix. The outlines of the shapes 'recovered' in this way are shown in Figure 16-4 below.

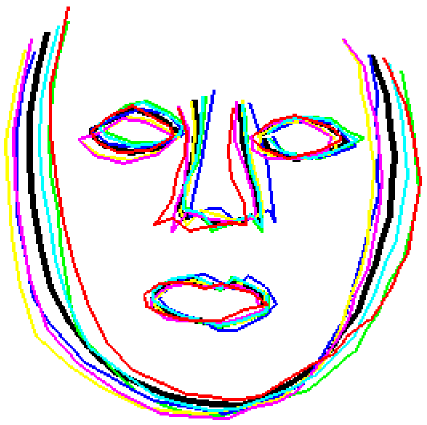

Figure 16-4 Reconstructed Shapes using 3 Principal Eigenvectors