Statistical Models of Appearance for Computer Vision [1]

T.F. Cootes and C.J.Taylor
Wolfson Image Analysis Unit,
Imaging Science and Biomedical Engineering,
University of Manchester,
Manchester M13 9PT, U.K.
email: `t.cootes@man.ac.uk`
http://www.wiau.man.ac.uk

July 10, 2000

[1]This is the first draft of a report describing our work on Active Shape Models and Active Appearance Models. Hopefully it will be expanded to become more comprehensive, when I get the time. My apologies for the missing and incomplete sections and the inconsistancies in notation. TFC

Contents

Chapter 1

Overview

The ultimate goal of machine vision is image understanding - the ability not only to recover image structure but also to know what it represents. By definition, this involves the use of models which describe and label the expected structure of the world. Over the past decade, model-based vision has been applied successfully to images of man-made objects. It has proved much more difficult to develop model-based approaches to the interpretation of images of complex and variable structures such as faces or the internal organs of the human body (as visualised in medical images). In such cases it has even been problematic to recover image structure reliably, without a model to organise the often noisy and incomplete image evidence. The key problem is that of variability. To be useful, a model needs to be specific - that is, to be capable of representing only 'legal' examples of the modelled object(s). It has proved difficult to achieve this whilst allowing for natural variability. Recent developments have overcome this problem; it has been shown that specific patterns of variability inshape and grey-level appearance can be captured by statistical models that can be used directly in image interpretation.

This document describes how one can build such models of shape and appearance, and how such models can be used to interpret images.

Chapter 2

Introduction

The majority of tasks to which machine vision might usefully be applied are 'hard'. The examples we use in this work are from medical image interpretation and face recognition, though the same considerations apply to many other domains. The most obvious reason for the degree of difficulty is that most non-trivial applications involve the need for an automated system to 'understand' the images with which it is presented - that is, to recover image structure and know what it means. This necessarily involves the use of models which describe and label the expected structure of the world. Real applications are also typically characterised by the need to deal with complex and variable structure - faces are a good example - and with images that provide noisy and possibly incomplete evidence - medical images are a good example, where it is often impossible to interpret a given image without prior knowledge of anatomy.

Model-based methods offer potential solutions to all these difficulties. Prior knowledge of the problem can, in principle, be used to resolve the potential confusion caused by structural complexity, provide tolerance to noisy or missing data, and provide a means of labelling the recovered structures. We would like to apply knowledge of the expected shapes of structures, their spatial relationships, and their grey-level appearance to restrict our automated system to 'plausible' interpretations. Of particular interest are generative models - that is, models sufficiently complete that they are able to generate realistic images of target objects. An example would be a face model capable of generating convincing images of any individual, changing their expression and so on. Using such a model, image interpretation can be formulated as a matching problem: given an image to interpret, structures can be located and labelled by adjusting the model's parameters in such a way that it generates an 'imagined image' which is as similar as possible to the real thing.

Because real applications often involve dealing with classes of objects which are not identical - for example faces - we need to deal with variability. This leads naturally to the idea of deformable models - models which maintain the essential characteristics of the class of objects they represent, but which can deform to fit a range of examples. There are two main characteristics we would like such models to possess. First, they should be general - that is, they should be capable of generating any plausible example of the class they represent. Second, and crucially, they should be specific - that is, they should only be capable of generating 'legal' examples - because, as we noted earlier, the whole point of using a model-based approach is to limit the attention of our system to plausible interpretations. In order to obtain specific models of variable objects, we need to acquire knowledge of how they vary.

Model-based methods make use of a prior model of what is expected in the image, and typically attempt to find the best match of the model to the data in a new image. Having matched the model, one can then make measurements or test whether the target is actually present.

This approach is a 'top-down' strategy, and differs significantly from 'bottom-up' or 'data-driven' methods. In the latter the image data is examined at a low level, looking for local structures such as edges or regions, which are assembled into groups in an attempt to identify objects of interest. Without a global model of what to expect, this approach is difficult and prone to failure.

A wide variety of model based approaches have been explored (see the review below). This work will concentrate on a statistical approach, in which a model is built from analysing the appearance of a set of labelled examples. Where structures vary in shape or texture, it is possible to learn what are plausible variations and what are not. A new image can be interpretted by finding the best plausible match of the model to the image data. The advantages of such a method are that

- It is widely applicable. The same algorithm can be applied to many different problems, merely by presenting different training examples.
- Expert knowledge can be captured in the system in the annotation of the training examples.
- The models give a compact representation of allowable variation, but are specific enough not to allow arbitrary variation different from that seen in the training set.
- The system need make few prior assumptions about the nature of the objects being modelled, other than what it learns from the training set. (For instance, there are no boundary smoothness parameters to be set.)

The models described below require a user to be able to mark 'landmark' points on each of a set of training images in such a way that each landmark represents a distinguishable point present on every example image. For instance, when building a model of the appearance of an eye in a face image, good landmarks would be the corners of the eye, as these would be easy to identify and mark in each image. This constrains the sorts of applications to which the method can be applied - it requires that the topology of the object cannot change and that the object is not so amorphous that no distinct landmarks can be applied. Unfortunately this rules out such things as cells or simple organisms which exhibit large changes in shape.

This report is in two main parts. The first describes building statistical models of shape and appearance. The second describes how these models can be used to interpret new images. This involves minimising a cost function defining how well a particular instance of a model describes the evidence in the image. Two approaches are described. The first, Active Shape Models, manipulates a shape model to describe the location of structures in a target image. The second, Active Appearance Models (AAMs), manipulate a model cabable of synthesising new images of the object of interest. The AAM algorithm seeks to find the model parameters which generate a synthetic image as close as possible to the target image. In each case the parameters of the best fitting model instance can be used for further processing, such as for measurement or classification.

Chapter 3

Background

The simplest model of an object is to use a typical example as a 'golden image'. A correlation method can be used to match (or register) the golden image to a new image. If structures in the golden image have been labelled, this match then gives the approximate position of the structures in the new image. For instance, one can determine the approximate locations of many structures in an MR image of a brain by registering a standard image, where the standard image has been suitably annotated by human experts. However, the variability of both shape and texture of most targets limits the precision of this method.

One approach to representing the variations observed in an image is to 'hand-craft' a model to solve the particular problem currently addressed. For instance Yuille *et al* [73] build up a model of a human eye using combinations of parameterised circles and arcs. Though this can be effective it is complicated, and a completely new solution is required for every application.

Staib and Duncan [68] represent the shapes of objects in medical images using fourier descriptors of closed curves. The choice of coefficients affects the curve complexity. Placing limits on each coefficient constrains the shape somewhat but not in a systematic way. It can be shown that such fourier models can be made directly equivalent to the statistical models described below, but are not as general. For instance, they cannot easily represent open boundaries.

Kass *et al* [41] introduced Active Contour Models (or 'snakes') which are energy minimising curves. In the original formulation the energy has an internal term which aims to impose smoothness on the curve, and an external term which encourages movement toward image features. They are particularly useful for locating the outline of general amorphous objects. However, since no model (other than smoothness) is imposed, they are not optimal for locating objects which have a known shape.

Alternative statistical approaches are described by Grenander et al [29] and Mardia et al [48]. These are, however, difficult to use in automated image interpretation. Goodall [27] and Bookstein [5] use statistical techniques for morphometric analysis, but do not address the problem of automated interpretation. Kirby and Sirovich [43] describe statistical modelling of grey-level appearance (particularly for face images) but do not address shape variability.

A more comprehensive survey of deformable models used in medical image analysis is given in [50].

Chapter 4

Statistical Shape Models

Here we describe the statistical models of shape which will be used to represent objects in images. The shape of an object is represented by a set of n points, which may be in any dimension. Commonly the points are in two or three dimensions. Shape is usually defined as that quality of a configuration of points which is invariant under some transformation. In two or three dimensions we usually consider the Euclidean transformation (translation, rotation and scaling). The shape of an object is not changed when it is moved, rotated or scaled.

Recent advances in the statistics of shape allow formal statistical techniques to be applied to sets of shapes, making possible analysis of shape differences and changes [20].

Our aim is to derive models which allow us to both analyse new shapes, and to synthesise shapes similar to those in a training set. The training set typically comes from hand annotation of a set of training images, though automatic landmarking systems are being developed (see below). By analysing the variations in shape over the training set, a model is built which can mimic this variation.

Much of the following will describe building models of shape in an arbitrary d-dimensional space, under a similarity transform T_θ (where θ are the parameters of the transformation). Most examples will be given for two dimensional shapes under the Euclidean transformation (with parameters of translation, scaling and orientation), as these are the easiest to represent on the page, and probably the most widely studied.

Note however that the dimensions need not always be in space, they can equally be time or intensity in an image. For instance

3D Shapes can either be composed of points in 3D space, or could be points in 2D with a time dimension (for instance in an image sequence)

2D Shapes can either be composed of points in 2D space, or one space and one time dimension

1D Shapes can either be composed of points along a line, or, as is used below, intensity values sampled at particular positions in an image.

There an numerous other possibilities. In each case a suitable transformation must be defined (eg Euclidean for 2D or global scaling and offset for 1D).

4.1 Suitable Landmarks

Good choices for landmarks are points which can be consistently located from one image to another. The simplest method for generating a training set is for a human expert to annotate each of a series of images with a set of corresponding points. In practice this can be very time consuming, and automatic and semi- automatic methods are being developed to aid this annotation.

In two dimensions points could be placed at clear corners of object boundaries, 'T' junctions between boundaries or easily located biological landmarks. However, there are rarely enough of such points to give more than a sparse desription of the shape of the target object. This list would be augmented with points along boundaries which are arranged to be equally spaced between well defined landmark points (Figure 4.1).

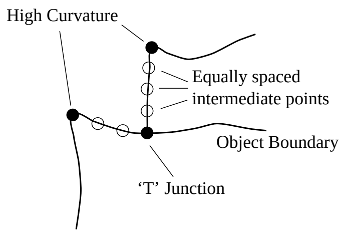


Figure 4.1: Good landmarks are points of high curvature or junctions. Intermediate points can be used to define boundary more precisely.

If a shape is described n points in d dimensions we represent the shape by a nd element vector formed by concatenating the elements of the individual point position vectors.

For instance, in a 2-D image we can represent the n landmark points, $\{(x_i, y_i)\}$, for a single example as the $2n$ element vector, $\mathbf{x}$, where

$$\mathbf{x} = (x_1, \ldots, x_n, y_1, \ldots, y_n)^T \tag{4.1}$$

Given s training examples, we generate s such vectors $\mathbf{x}_j$. Before we can perform statistical analysis on these vectors it is important that the shapes represented are in the same co-ordinate frame. We wish to remove variation which could be attributable to the allowed global transformation, T.

4.2 Aligning the Training Set

There is considerable literature on methods of aligning shapes into a common co-ordinate frame, the most popular approach being Procrustes Analysis [27]. This aligns each shape so that the sum of distances of each shape to the mean ($D = \sum |\mathbf{x}_i - \bar{\mathbf{x}}|^2$) is minimised. It is poorly defined unless constraints are placed on the alignment of the mean (for instance, ensuring it is centred on the origin, has unit scale and some fixed but arbitrary orientation). Though analytic solutions exist to the alignment of a set, a simple iterative approach is as follows:

1. Translate each example so that its centre of gravity is at the origin.
2. Choose one example as an initial estimate of the mean shape and scale so that $|\bar{\mathbf{x}}| = 1$.
3. Record the first estimate as $\bar{\mathbf{x}}_0$ to define the default reference frame.
4. Align all the shapes with the current estimate of the mean shape.
5. Re-estimate mean from aligned shapes.
6. Apply constraints on the current estimate of the mean by aligning it with $\bar{\mathbf{x}}_0$ and scaling so that $|\bar{\mathbf{x}}| = 1$.
7. If not converged, return to **4**.

   (Convergence is declared if the estimate of the mean does not change significantly after an iteration)

The operations allowed during the alignment will affect the shape of the final distribution. For two and three dimensional shapes a common approach is to centre each shape on the origin, scale each so that $|\mathbf{x}| = 1$ and then choose the orientation for each which minimises D. The scaling constraint means that the aligned shapes $\mathbf{x}$ lie on a hypersphere, which can introduce significant non-linearities if large shape changes occur. For instance, Figure 4.2(a) shows the corners of a set of rectangles with varying aspect ratio (a linear change), aligned in this fashion. The scale constraint ensures all the corners lie on a circle about the origin. A linear change in the aspect ratio introduces a non-linear variation in the point positions. If we can arrange that the points lie closer to a straight line, it simplifies the description of the distribution used later in the analysis.

An alternative approach is to allow both scaling and orientation to vary when minimising D. Suppose $T_{s,\theta}(\mathbf{x})$ scales the shape, $\mathbf{x}$, by s and rotates it by θ.

To align two 2D shapes, $\mathbf{x}_1$ and $\mathbf{x}_2$, each centred on the origin ($\mathbf{x}_1.\mathbf{1} = \mathbf{x}_2.\mathbf{1} = 0$), we choose a scale, s, and rotation, θ, so as to minimise $|T_{s,\theta}(\mathbf{x}_1) - \mathbf{x}_2|^2$, the sum of square distances between points on shape $\mathbf{x}_2$ and those on the scaled and rotated version of shape $\mathbf{x}_1$. Appendix B gives the optimal solution.

If this approach is used to align the set of rectangles, Figure 4.2(b), their corners lie on circles offset from the origin. This introduces even greater non-linearity than the first approach.

A third approach is to transform each shape into the *tangent space* to the mean so as to minimise D. The tangent space to $\mathbf{x}_t$ is the hyperplane of vectors normal to $\mathbf{x}_t$, passing through $\mathbf{x}_t$. ie All the vectors $\mathbf{x}$ such that $(\mathbf{x}_t - \mathbf{x}).\mathbf{x}_t = 0$, or $\mathbf{x}.\mathbf{x}_t = 1$ if $|\mathbf{x}_t| = 1$. Figure 4.2(c) demonstrates that for the rectangles this leads to the corners varying along a straight lines, orthogonal to the lines from the origin to the corners of the mean shape (a square). This preserves the linear nature of the shape variation. The simplest way to achieve this is to align the shapes with the mean, allowing scaling and rotation, then project into the tangent space by scaling $\mathbf{x}$ by $1/(\mathbf{x}.\bar{\mathbf{x}})$.

Different approaches to alignment can produce different distributions of the aligned shapes. We wish to keep the distribution compact and keep any non-linearities to a minimum, so use the tangent space approach in the following.

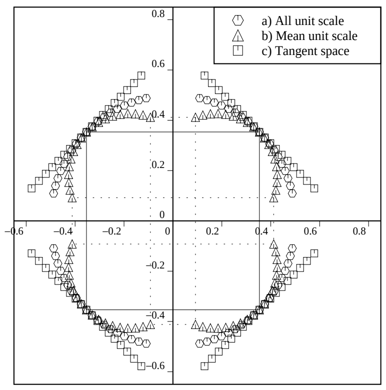


Figure 4.2: Aligning rectangles with varying aspect ration. a) All shapes set to unit scale, b) Scale and angle free, c) Align into tangent space

4.3 Modelling Shape Variation

Suppose now we have s sets of points $\mathbf{x}_i$ which are aligned into a common co-ordinate frame. These vectors form a distribution in the nd dimensional space in which they live. If we can model this distribution, we can generate new examples, similar to those in the original training set, and we can examine new shapes to decide whether they are plausible examples.

In particular we seek a parameterised model of the form $\mathbf{x} = M(\mathbf{b})$, where $\mathbf{b}$is a vector of parameters of the model. Such a model can be used to generate new vectors, $\mathbf{x}$. If we can model the distribution of parameters, $p(\mathbf{b}$ we can limit them so that the generated $\mathbf{x}$'s are similar to those in the training set. Similarly it should be possible to estimate $p(\mathbf{x})$ using the model.

To simplify the problem, we first wish to reduce the dimensionality of the data from nd to something more manageable. An effective approach is to apply Principal Component Analysis (PCA) to the data. The data form a cloud of points in the nd-D space. PCA computes the main axes of this cloud, allowing one to approximate any of the original points using a model with fewer than nd parameters. The approach is as follows.

1. Compute the mean of the data,

$$\bar{\mathbf{x}} = \frac{1}{s}\sum_{i=1}^{s}\mathbf{x}_i \tag{4.2}$$

2. Compute the covariance of the data,

$$\mathbf{S} = \frac{1}{s-1}\sum_{i=1}^{s}(\mathbf{x}_i - \bar{\mathbf{x}})(\mathbf{x}_i - \bar{\mathbf{x}})^T \tag{4.3}$$

3. Compute the eigenvectors, ϕ_i and corresponding eigenvalues λ_i of $\mathbf{S}$ (sorted so that $\lambda_i \geq \lambda_{i+1}$). When there are fewer samples than dimensions in the vectors, there are quick methods of computing these eigenvectors - see Appendix A.

If $\mathbf{\Phi}$ contains the t eigenvectors corresponding to the largest eigenvalues, then we can then approximate any of the training set, $\mathbf{x}$ using

$$\mathbf{x} \approx \bar{\mathbf{x}} + \mathbf{\Phi}\mathbf{b} \tag{4.4}$$

where $\mathbf{\Phi} = (\phi_1|\phi_2|\ldots|\phi_t)$ and $\mathbf{b}$ is a t dimensional vector given by

$$\mathbf{b} = \mathbf{\Phi}^T(\mathbf{x} - \bar{\mathbf{x}}) \tag{4.5}$$

The vector $\mathbf{b}$ defines a set of parameters of a deformable model. By varying the elements of $\mathbf{b}$ we can vary the shape, $\mathbf{x}$using Equation 4.4. The variance of the i^{th} parameter, b_i, across the training set is given by λ_i. By applying limits of $\pm 3\sqrt{\lambda_i}$ to the parameter b_i we ensure that the shape generated is similar to those in the original training set.

The number of eigenvectors to retain, t, can be chosen so that the model represents some proportion (eg 98%) of the total variance of the data, or so that the residual terms can be considered noise. See section 4.4 below.

For instance, Figure 4.3 shows the principal axes of a 2D distribution of vectors. In this case any of the points can be approximated by the nearest point on the principal axis through the mean. $\mathbf{x} \approx \mathbf{x}' = \bar{\mathbf{x}} + b\mathbf{p}$ where b is the distance along the axis from the mean of the closest approach to $\mathbf{x}$. Thus the two dimensional data is approximated using a model with a single parameter, b. Similarly shape models controlling many hundreds of model points may need only a few parameters to approximate the examples in the original training set.

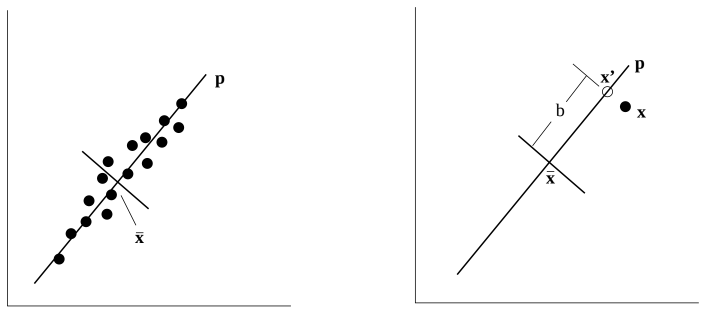


Figure 4.3: Applying a PCA to a set of 2D vectors. $\mathbf{p}$ is the principal axis. Any point $\mathbf{x}$can be approximated by the nearest point on the line, $\mathbf{x}'$ (see text).

4.4 Choice of Number of Modes

The number of modes to retain, t, can be chosen in several ways. Probably the simplest is to choose t so as to explain a given proportion (eg 98%) of the variance exhibited in the training set.

Let λ_i be the eigenvalues of the covariance matrix of the training data. Each eigenvalue gives the variance of the data about the mean in the direction of the corresponding eigenvector. The total variance in the training data is the sum of all the eigenvalues, $V_T = \sum \lambda_i$.

We can then choose the t largest eigenvalues such that

$$\sum_{i=1}^{t} \lambda_i \geq f_v V_T \tag{4.6}$$

where f_v defines the proportion of the total variation one wishes to explain (for instance, 0.98 for 98%).

If the noise on the measurements of the (aligned) point positions has a variance of σ_n^2, then we could choose the largest t such that $\lambda_t > \sigma_n^2$, assuming that the eigenvalues are sorted into descending order.

An alternative approach is to choose enough modes that the model can approximate any training example to within a given accuracy. For instance, we may wish that the best approximation to an example has every point within one pixel of the corresponding example points.

To achieve this we build models with increasing numbers of modes, testing the ability of each to represent the training set. We choose the first model which passes our desired criteria.

Additional confidence can be obtained by performing this test in a miss-one-out manner. We choose the smallest t for the full model such that models built with t modes from all but any one example can approximate the missing example sufficiently well.

4.5 Examples of Shape Models

Figure 4.4 shows the outlines of a hand used to train a shape model. They were obtained from a set of images of the authors hand. Each is represented by 72 landmark points. The ends and junctions of the fingers are true landmarks, other points were equally spaced along the boundary between.

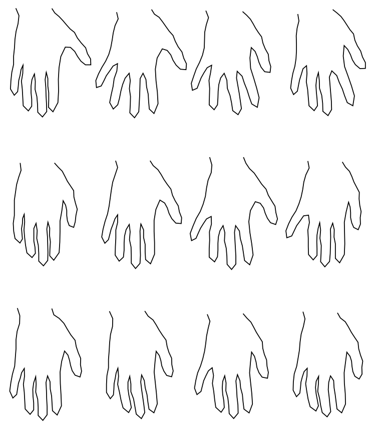


Figure 4.4: Example shapes from training set of hand outlines

Building a shape model from 18 such examples leads to a model of hand shape variation whose modes are demonstrated in figure 4.5.

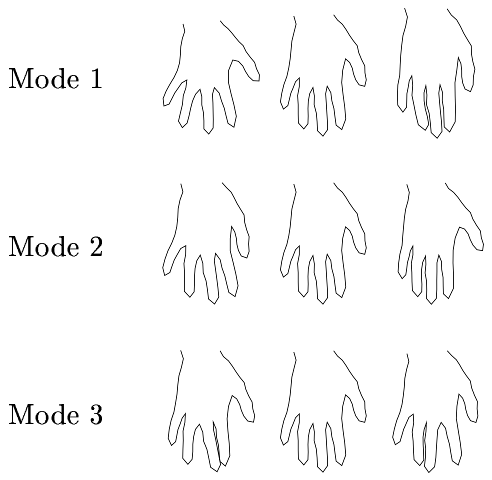


Figure 4.5: Effect of varying each of first three hand model shape parameters in turn between ± 3 s.d.

Images of the face can demonstrate a wide degree of variation in both shape and texture. Appearance variations are caused by differences between individuals, the deformation of an individual face due to changes in expression and speaking, and variations in the lighting. Typically one would like to locate the features of a face in order to perform further processing (see Figure 4.6). The ultimate aim varies from determining the identity or expression of the person to deciding in which direction they are looking [47].

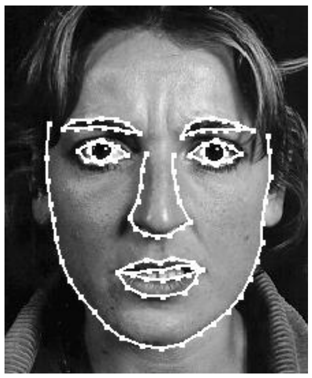


Figure 4.6: Example face image annotated with landmarks

Figure 4.7 shows example shapes from a training set of 300 labelled faces (see Figure 4.6 for an example image showing the landmarks). Each image is annotated with 133 landmarks. The shape model has 36 modes, which explain 98% of the variance in the landmark positions in the training set. Figure 4.8 shows the effect of varying the first three shape parameters in turn between ± 3 standard deviations from the mean value, leaving all other parameters at zero. These modes explain global variation due to 3D pose changes, which cause movement of all the landmark points relative to one another. Less significant modes cause smaller, more local changes. The modes obtained are often similar to those a human would choose if designing a parameterised model. However, they are derived directly from the statistics of a training set and will not always separate shape variation in an obvious manner.

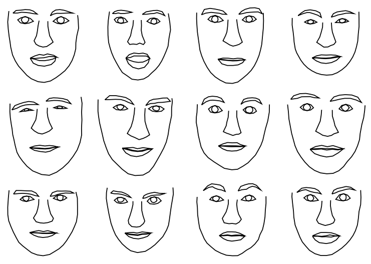


Figure 4.7: Example shapes from training set of faces

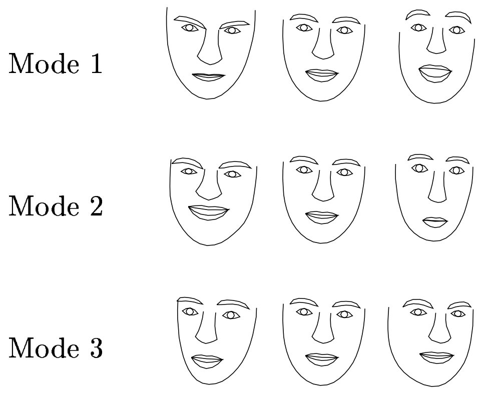


Figure 4.8: Effect of varying each of first three face model shape parameters in turn between ± 3 s.d.

4.6 Generating Plausible Shapes

If we wish to use the model $\mathbf{x} = \bar{\mathbf{x}} + \mathbf{\Phi b}$ to generate examples similar to the training set, we must choose the parameters, $\mathbf{b}$, from a distribution learnt from the training set. Thus we must estimate this distribution, $p(\mathbf{b})$, from the training set. We will define a set of parameters as 'plausible' if $p(\mathbf{b}) \geq p_t$, where p_t is some suitable threshold on the p.d.f.. p_t is usually chosen so that some proportion (eg 98%) of the training set passes the threshold.

If we assume that b_i are independent and gaussian, then

$$\log p(\mathbf{b}) = -0.5 \sum_{i=1}^{t} \frac{b_i^2}{\lambda_i} + const \tag{4.7}$$

To constrain $\mathbf{b}$ to plausible values we can either apply hard limits to each element, b_i, (for instance $|b_i| \le 3\sqrt{\lambda_i}$), or we can constrain $\mathbf{b}$ to be in a hyperellipsoid,

$$\left(\sum_{i=1}^{t} \frac{b_i^2}{\lambda_i} \right) \le M_t \tag{4.8}$$

Where the threshold, M_t, is chosen using the χ^2 distribution.

4.6.1 Non-Linear Models for PDF

Approximating the distribution as a gaussian (or as uniform in a hyper-box) works well for a wide variety of examples, but cannot adequately represent non-linear shape variations, such as those generated when parts of the object rotate, or there are changes in viewing position of a 3D object. There have been several non-linear extensions to the PDM, either using polynomial modes [66], using a multi-layer perceptron to perform non-linear PCA [65] or using polar co-ordinates for rotating sub-parts of the model [31].

However, all these approaches assume that varying the parameters $\mathbf{b}$ within given limits will always generate plausible shapes, and that all plausible shapes can be so generated. This is not always the case. For instance, if a sub-part of the shape can appear in one of two positions, but not in-between, then the distribution has two separate peaks, with an illegal space in between. Without imposing more complex constraints on the parameters $\mathbf{b}$, models of the form $\mathbf{x} = f(\mathbf{b})$ are likely to generate illegal shapes.

For instance, consider the set of synthetic training examples shown in Figure 4.9. Here 28 points are used to represent a triangle rotating inside a square (there are 3 points along each line segment). If we apply PCA to the data, we find there are two significant components. Projecting the 100 original shapes $\mathbf{x}$ into the 2-D space of $\mathbf{b}$ (using (4.5)) gives the distribution shown in Figure 4.10. This is clearly not gaussian. To generate new examples using the model which are similar to the training set we must constrain the parameters $\mathbf{b}$ to be near the edge of the circle. Points at the mean ($\mathbf{b} = \mathbf{0}$) should actually be illegal.

One approach would be to use an alternative parameterisation of the shapes. Heap and Hogg [31] use polar coordinates for some of the model points, relative to other points. A more general approach is to use non-linear models of the probability density function, $p(\mathbf{b})$. This allows the modelling of distinct classes of shape as well as non-linear shape variation, and does not require any labelling of the class of each training example.

A useful approach is to model $p(\mathbf{b})$ using a mixture of gaussians approximation to a kernel density estimate of the distribution.

$$p_{mix}(\mathbf{x}) = \sum_{j=1}^{m} w_j G(\mathbf{x} : \mu_j, \mathbf{S}_j) \tag{4.9}$$

See Appendix H for details.

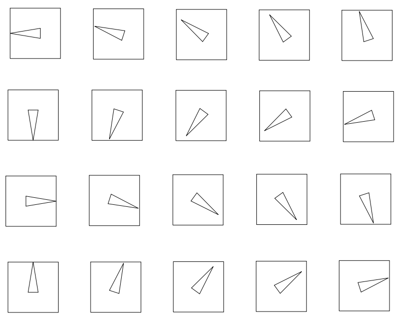


Figure 4.9: Examples from training set of synthetic shapes

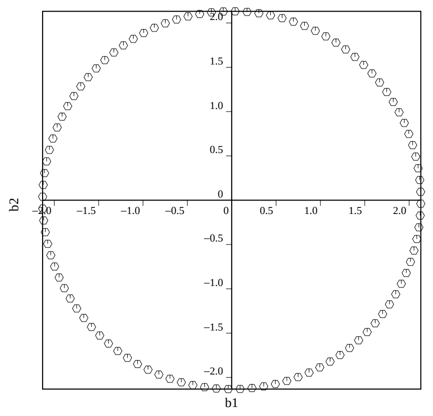


Figure 4.10: Distribution of **b**for 100 synthetic shapes

Example of the PDF for a Set of Shapes

Figure 4.11 shows the p.d.f. estimated for **b** for the rotating triangle set described above. The adaptive kernel method was used, with the initial h estimated using cross-validation. The desired number of components can be obtained by specifying an acceptable approximation error. Figure 4.12 shows the estimate of the p.d.f. obtained by fitting a mixture of 12 gaussians to the data.

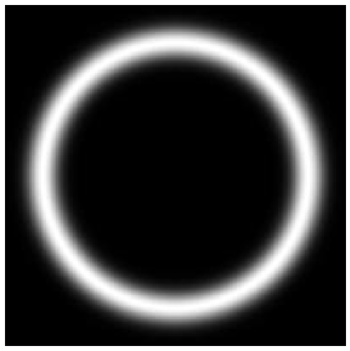


Figure 4.11: Plot of pdf estimated using the adaptive kernel method

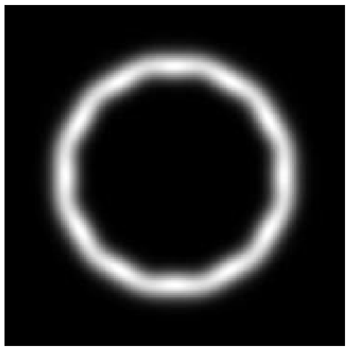


Figure 4.12: Plot of pdf approximation using mixture of 12 gaussians

4.7 Finding the Nearest Plausible Shape

When fitting a model to a new set of points, we have the problem of finding the nearest plausible shape, $\mathbf{x}$ to a target shape, $\mathbf{x}$'. The first estimate is to project into the parameter space, giving $\mathbf{b}' = \mathbf{\Phi}^T(\mathbf{x}' - \bar{\mathbf{x}})$.

We define a set of parameters as plausible if $p(\mathbf{b}) \geq p_t$. If $p(\mathbf{b}) < p_t$ we wish to move $\mathbf{x}$ to the nearest point at which it is considered plausible.

If our model of $p(\mathbf{b})$ is as a single gaussian, we can simply truncate the elements b_i such that $|b_i| \leq 3\sqrt{\lambda_i}$. Alternatively we can scale $\mathbf{b}$ until

$$\left(\sum_{i=1}^{t} \frac{b_i^2}{\lambda_i} \right) \leq M_t \tag{4.10}$$

Where the threshold, M_t, is chosen using the χ^2 distribution.

If, however, we use a mixture model to represent $p(\mathbf{b})$, and $p(\mathbf{b}') < p_t$, we must find the nearest $\mathbf{b}$ such that $p(\mathbf{b}) \geq p_t$. In practice this is difficult to locate, but an acceptable approximation can be obtained by gradient ascent - simply move uphill until the threshold is reached. The gradient of (4.9) is straightforward to compute, and suitable step sizes can be estimated from the distance to the mean of the nearest mixture component.

For instance, Figure 4.13 shows an example of the synthetic shape used above, with its points perturbed by noise. Figure 4.14 shows the result of projecting into the space of $\mathbf{b}$ and back. There is significant reduction in noise, but the triangle is unacceptably large compared with examples in the training set. Figure 4.15 shows the shape obtained by gradient ascent to the nearest plausible point using the 12 component mixture model estimate of $p(\mathbf{b})$. The triangle is now similar in scale to those in the training set.

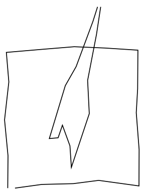


Figure 4.13: Shape with noise

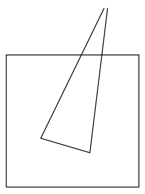


Figure 4.14: Projection into $\mathbf{b}$-space

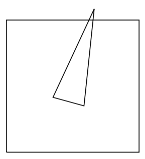


Figure 4.15: Nearby plausible shape

4.8 Fitting a Model to New Points

An example of a model in an image is described by the shape parameters, $\mathbf{b}$, combined with a transformation from the model co-ordinate frame to the image co-ordinate frame. Typically this will be a euclidean transformation defining the position, (X_t, Y_t), orientation, θ, and scale, s, of the model in the image.

The positions of the model points in the image, $\mathbf{x}$, are then given by

$$\mathbf{x} = T_{X_t,Y_t,s,\theta}(\bar{\mathbf{x}} + \mathbf{\Phi}\mathbf{b}) \tag{4.11}$$

Where the function $T_{X_t,Y_t,s,\theta}$ performs a rotation by θ, a scaling by s and a translation by (X_t, Y_t). For instance, if applied to a single point (xy),

$$T_{X_t,Y_t,s,\theta}\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} X_t \\ Y_t \end{pmatrix} + \begin{pmatrix} s\cos\theta & s\sin\theta \\ -s\sin\theta & s\cos\theta \end{pmatrix}\begin{pmatrix} x \\ y \end{pmatrix} \tag{4.12}$$

Suppose now we wish to find the best pose and shape parameters to match a model instance $\mathbf{x}$to a new set of image points, $\mathbf{Y}$. Minimising the sum of square distances between corresponding model and image points is equivalent to minimising the expression

$$|\mathbf{Y} - T_{X_t,Y_t,s,\theta}(\bar{\mathbf{x}} + \mathbf{\Phi}\mathbf{b})|^2 \tag{4.13}$$

A simple iterative approach to achieving this is as follows:

1. Initialise the shape parameters, $\mathbf{b}$, to zero
2. Generate the model instance $\mathbf{x} = \bar{\mathbf{x}} + \mathbf{\Phi}\mathbf{b}$
3. Find the pose parameters (X_t, Y_t, s, θ) which best map $\mathbf{x}$to $\mathbf{Y}$ (See Appendix B).
4. Invert the pose parameters and use to project $\mathbf{Y}$ into the model co-ordinate frame:
$$\mathbf{y} = T^{-1}_{X_t,Y_t,s,\theta}(\mathbf{Y}) \tag{4.14}$$
5. Project $\mathbf{y}$ into the tangent plane to $\bar{\mathbf{x}}$ by scaling by $1/(\mathbf{y}.\bar{\mathbf{x}})$.
6. Update the model parameters to match to $\mathbf{y}$
$$\mathbf{b} = \mathbf{\Phi}^T(\mathbf{y} - \bar{\mathbf{x}}) \tag{4.15}$$
7. Apply constraints on $\mathbf{b}$(see 4.6,4.7 above).
8. If not converged, return to step 2.

Convergence is declared when applying an iteration produces no significant change in the pose or shape parameters. This approach usually converges in a few iterations.

4.9 Testing How Well the Model Generalises

The shape models described use linear combinations of the shapes seen in a training set. In order to be able to match well to a new shape, the training set must exhibit all the variation expected in the class of shapes being modelled. If it does not, the model will be over-constrained and will not be able to match to some types of new example. For instance, a model trained only on squares will not generalise to rectangles.

One approach to estimating how well the model will perform is to use jack-knife or miss-one-out experiments. Given a training set of s examples, build a model from all but one, then fit the model to the example missed out and record the error (for instance using (4.13)). Repeat this, missing out each of the s examples in turn. If the error is unacceptably large for any example, more training examples are probably required. However, small errors for all examples only mean that there is more than one example for each type of shape variation, not that all types are properly covered (though it is an encouraging sign).

Equation (4.13) gives the sum of square errors over all points, and may average out large errors on one or two individual points. It is often wise to calculate the error for each point and ensure that the maximum error on any point is sufficiently small.

4.10 Estimating $p(shape)$

Given a configuration of points, $\mathbf{x}$, we would like to be able to decide whether $\mathbf{x}$ is a plausible example of the class of shapes described by our training set.

The original training set, once aligned, can be considered a set of samples from a probability density function, $p(\mathbf{x})$, which we must estimate.

Any shape can be approximated by the nearest point in the sub-space defined by the eigenvectors, $\mathbf{\Phi}$. A point in this subspace is defined by a vector of shape parameters, $\mathbf{b}$.

Let $d\mathbf{x} = \mathbf{x} - \bar{\mathbf{x}}$. Then the best (least squares) approximation is given by $\mathbf{x}' = \mathbf{\Phi b}$ where $\mathbf{b} = \mathbf{\Phi}^T d\mathbf{x}$.

The residual error is then $\mathbf{r} = d\mathbf{x} - \mathbf{\Phi b}$.

The square magnitude of this is

$$\begin{aligned} |\mathbf{r}|^2 &= \mathbf{r}^T\mathbf{r} \\ &= d\mathbf{x}^T d\mathbf{x} - 2d\mathbf{x}\mathbf{\Phi b} + \mathbf{b}^T\mathbf{\Phi}^T\mathbf{\Phi b} \\ |\mathbf{r}|^2 &= |d\mathbf{x}|^2 - |\mathbf{b}|^2 \end{aligned} \tag{4.16}$$

Applying a PCA splits the shape vector into two orthogonal components with coefficients described by the elements of $\mathbf{b}$and $\mathbf{r}$, which we assume to be independent.

Thus

$$p(\mathbf{x}) = p(\mathbf{r}).p(\mathbf{b}) \tag{4.17}$$

$$\log p(\mathbf{x}) = \log p(\mathbf{r}) + \log p(\mathbf{b}) \tag{4.18}$$

If we assume that each element of $\mathbf{r}$ is independent and distributed as a gaussian with variance σ_r^2, then

$$\begin{aligned} p(\mathbf{r}) &\propto \exp(-0.5|\mathbf{r}|^2/\sigma_r^2) \\ \log p(\mathbf{r}) &= -0.5|\mathbf{r}|^2/\sigma_r^2 + const \end{aligned} \tag{4.19}$$

The distribution of parameters, $p(\mathbf{b})$, can be estimated as described in previous sections. Given this, we can estimate the p.d.f.at a new shape, $\mathbf{x}$, using

$$\log p(\mathbf{x}) = \log p(\mathbf{b}) - 0.5(|d\mathbf{x}|^2 - |\mathbf{b}|^2)/\sigma_r^2 + const \tag{4.20}$$

The value of σ_r^2 can be estimated from miss-one-out experiments on the training set.

4.11 Relaxing Shape Models

When only a few examples are available in a training set, the model built from them will be overly constrained - only variation observed in the training set is represented.

It is possible to artificially add extra variation, allowing more flexibility in the model.

4.11.1 Finite Element Models

Finite Element Methods allow us to take a single shape and treat it as if it were made of an elastic material. The techniques of Modal Analysis give a set of linear deformations of the shape equivalent to the resonant modes of vibration of the original shape. Such an approach has been used by several groups to develop deformable models for computer vision, including Pentland and Sclaroff [55], Karaolani *et. al.*[40] and Nastar and Ayache [53]. However, the modes are somewhat arbitrary and may not be representative of the real variations which occur in a class of shapes.

A elastic body can be represented as a set of n nodes, a mass matrix $\mathbf{M}$ and a stiffness matrix $\mathbf{K}$. In two dimensions these are both $2n \times 2n$. Modal Analysis allows calculation of a set of vibrational modes by solving the generalised eigenproblem

$$\mathbf{K}\mathbf{\Phi}_v = \mathbf{M}\mathbf{\Phi}_v\mathbf{\Omega}^2 \tag{4.21}$$

where $\mathbf{\Phi}_v$ is a matrix of eigenvectors representing the modes and $\mathbf{\Omega}^2 = diag(\omega_1^2, \ldots, \omega_n^2)$ is a diagonal matrix of eigenvalues. (ω_i is the frequency of the i^{th} mode). The energy of deformation in the i^{th} mode is proportional to ω_i^2.

If we assume the structure can be modelled as a set of point unit masses the mass matrix, $\mathbf{M}$, becomes the identity, and (4.21) simplifies to computing the eigenvectors of the symmetric matrix $\mathbf{K}$,

$$\mathbf{K}\mathbf{\Phi}_v = \mathbf{\Phi}_v\mathbf{\Omega}^2 \tag{4.22}$$

Thus if $\mathbf{u}$ is a vector of weights on each mode, a new shape can be generated using

$$\mathbf{x} = \bar{\mathbf{x}} + \mathbf{\Phi}_v\mathbf{u} \tag{4.23}$$

4.11.2 Combining Statistical and FEM Modes

Equation 4.23 is clearly related to the form of the statistical shape models described above. Both are linear models. This suggests that there is a way to combine the two approaches. If we have just one example shape, we cannot build a statistical model and our only option is the FEM approach to generating modes. If, however, we have two examples, we can build a statistical model, but it would only have a single mode, linearly interpolating between the two shapes. It would have no way of modelling other distortions.

One approach to combining FEMs and the statistical models is as follows. We calculate the modes of vibration of both shapes, then use them to generate a large number of new examples by randomly selecting model parameters $\mathbf{u}$ using some suitable distribution. We then train a statistical model on this new set of examples. The resulting model would then incorporate a mixture of the modes of vibration and the original statistical mode interpolating between the original shapes.

Such a strategy would be applicable for any number of shapes. However, we should decrease the magnitude of the allowed vibration modes as the number of examples increases to avoid incorporating spurious modes. As we get more training examples, we need rely less on the artificial modes generated by the FEM.

It would be time consuming and error prone to actually generate large numbers of synthetic examples from our training set. Fortunately the effect can be achieved with a little matrix algebra.

For simplicity and efficiency, we assume the modes of vibration of any example can be approximated by those of the mean.

Let $\bar{\mathbf{x}}$ be the mean of a set of s examples, $\mathbf{S}$ be the covariance of the set and $\mathbf{\Phi}_v$ be the modes of vibration of the mean derived by FEM analysis.

Suppose we were to generate a set of examples by selecting the values for $\mathbf{u}$ from a distribution with zero mean and covariance $\mathbf{S}_u$. The distribution of $\mathbf{\Phi}_v\mathbf{u}$ then has a covariance of

$$\mathbf{\Phi}_v\mathbf{S}_u\mathbf{\Phi}_v^T \tag{4.24}$$

and the distribution of $\mathbf{x} = \mathbf{x}_i + \mathbf{\Phi}_v\mathbf{u}$ will have a covariance of

$$\mathbf{C_i} = \mathbf{x}_i\mathbf{x}_i^T + \mathbf{\Phi}_v\mathbf{S}_u\mathbf{\Phi}_v^T \tag{4.25}$$

about the origin.

If we treat the elements of $\mathbf{u}$ as independent and normally distributed about zero with a variance on u_i of $\alpha\lambda_{ui}$, then

$$\mathbf{S}_u = \alpha\Lambda_u \tag{4.26}$$

where $\Lambda_u = diag(\lambda_{u1}\ldots)$.

The covariance of $\mathbf{x}_i$ about the origin is then

$$\mathbf{C}_i = \mathbf{x}_i\mathbf{x}_i^T + \alpha\mathbf{\Phi}_v\Lambda_u\mathbf{\Phi}_v^T \tag{4.27}$$

If the frequency associated with the j^{th} mode is ω_j then we will choose

$$\lambda_{uj} = \omega_j^{-2} \tag{4.28}$$

This gives a distribution which has large variation in the low frequency, large scale deformation modes, and low variation in the more local high frequency modes.

One can justify the choice of λ_{uj} by considering the strain energy required to deform the original shape, $\bar{\mathbf{x}}$, into a new example $\mathbf{x}$. The contribution to the total from the j^{th} mode is

$$E_j = \frac{1}{2}u_j^2\omega_j^2 \tag{4.29}$$

The form of Equation 4.29 ensures that the energy tends to be spread equally amonst all the modes.

The constant α controls the magnitude of the deformations, and is discussed below.

4.11.3 Combining Examples

To calculate the covariance about the origin of a set of examples drawn from distributions about m original shapes, $\{\mathbf{x}_i\}$, we simply take the mean of the individual covariances

$$\begin{aligned}\mathbf{s}' &= \frac{1}{m}\sum_{i=1}^{m}\left(\alpha\mathbf{\Phi}_v\Lambda_u\mathbf{\Phi}_v^T + \mathbf{x}_i\mathbf{x}_i^T\right) \\ &= \alpha\mathbf{\Phi}_v\Lambda_u\mathbf{\Phi}_v^T + \frac{1}{m}\sum_{i=1}^{m}\mathbf{x}_i\mathbf{x}_i^T \\ &= \alpha\mathbf{\Phi}_v\Lambda_u\mathbf{\Phi}_v^T + \mathbf{S}_m + \bar{\mathbf{x}}\bar{\mathbf{x}}^T\end{aligned} \tag{4.30}$$

where $\mathbf{S}_m$ is the covariance of the m original shapes about their mean.

Thus the covariance about the mean $\bar{\mathbf{x}}$ is then

$$\begin{aligned}\mathbf{S} &= \mathbf{S}' - \bar{\mathbf{x}}\bar{\mathbf{x}}^T \\ &= \mathbf{S}_m + \alpha\mathbf{\Phi}_v\Lambda_u\mathbf{\Phi}_v^T\end{aligned} \tag{4.31}$$

We can then build a combined model by computing the eigenvectors and eigenvalues of this matrix to give the modes. When $\alpha = 0$ (the magnitude of the vibrations is zero) $\mathbf{S} = \mathbf{S}_m$ and we get the result we would from a pure statistical model.

When we allow non-zero vibrations of each training example, $(\alpha > 0)$, the eigenvectors of $\mathbf{S}$ will include the effects of the vibrational modes.

As the number of examples increases we wish to rely more on the statistics of the real data and less on the artificial variation introduced by the modes of vibration. To achieve this we must reduce α as m increases. We use the relationship

$$\alpha = \alpha_1/m \tag{4.32}$$

4.11.4 Examples

Two sets of 16 points were generated, one forming a square, the other a rectangle with aspect ratio 0.5. Figure 4.16 shows the modes corresponding to the four smallest eigenvalues of the FEM governing equation for the square. Figure 4.17 shows those for a rectangle.

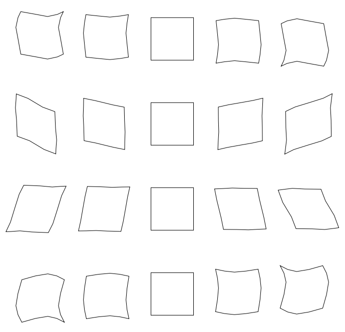


Figure 4.16: First four modes of vibration of a square

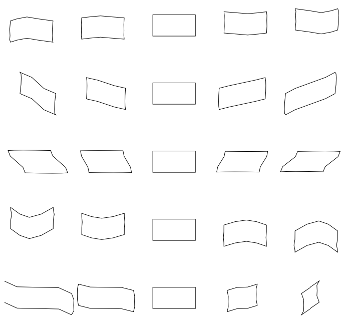


Figure 4.17: First four modes of vibration of a rectangle

Figure 4.18 shows the modes of variation generated from the eigenvectors of the combined covariance matrix (Equation 4.31). This demonstrates that the principal mode is now the mode which changes the aspect ratio (which would be the only one for a statistical model trained on the two shapes). Other modes are similar to those generated by the FEM analysis.

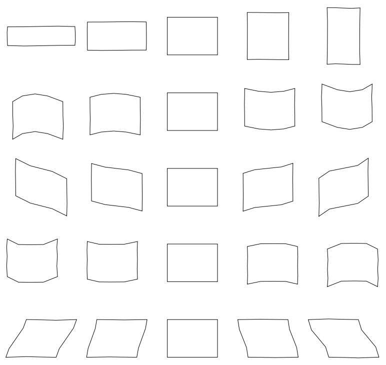


Figure 4.18: First modes of variation of a combined model containing a square and a rectangle. The mode controlling the aspect ratio is the most significant.

4.11.5 Relaxing Models with a Prior on Covariance

Rather than using Finite Element Methods to generate artificial modes, a similar effect can be achieved simply by adding extra values to elements of the covariance matrix during the PCA. This is equivalent to specifying a prior on the covariance matrix. The approach is to compute the covariance matrix of the data, then either add a small amount to the diagonal, or to the ij^{th} elements which correspond to covariance between ordinates of nearby points. Encouraging higher covariance between nearby points will generate artificial modes similar to the elastic modes of vibration. The magnitude of the addition should be inversely proportional to the number of samples available. For instance, see [15] or work by Wang and Staib [72].

Chapter 5

Statistical Models of Appearance

To synthesise a complete image of an object or structure, we must model both its shape and its texture (the pattern of intensity or colour across the region of the object). Here we describe how statistical models can be built to represent both shape variation, texture variation and the correllations between them. Such models can be used to generate photo-realistic (if necessary) synthetic images.

The models are generated by combining a model of shape variation with a model of the texture variations in a shape-normalised frame. By 'texture' we mean the pattern of intensities or colours across an image patch. We require a training set of labelled images, where key landmark points are marked on each example object. For instance, to build a face model we require face images marked with points at key positions to outline the main features (Figure 5.1).

Given such a set we can generate a statistical model of shape variation from the points (see Chapter 4 for details). Given a mean shape, we can warp each training example into the mean shape, to obtain a 'shape-free' patch (Figure 5.1). We then build a statistical model of the texture variation in this patch (essentially an eigen-face type model [70]).

There will be correlations between the parameters of the shape model and those of the texture model across the training set. To take account of these we build a combined appearance model which controls both shape and texture.

The following sections describe these steps in more detail.

5.1 Statistical Models of Texture

To build a statistical model of the texture (intensity or colour over an image patch) we warp each example image so that its control points match the mean shape (using a triangulation algorithm - see Appendix G). This removes spurious texture variation due to shape differences which would occur if we simply performed eigenvector decomposition on the un-normalised face patches (as in the eigen-face approach [70]). We then sample the intensity information from the *shape-normalised* image over the region covered by the mean shape to form a texture vector, $\mathbf{g}_{im}$. For example, Figure 5.1 shows a labelled face image, the model points and the face patch normalised into the mean shape. The sampled patch contains little of the texture variation

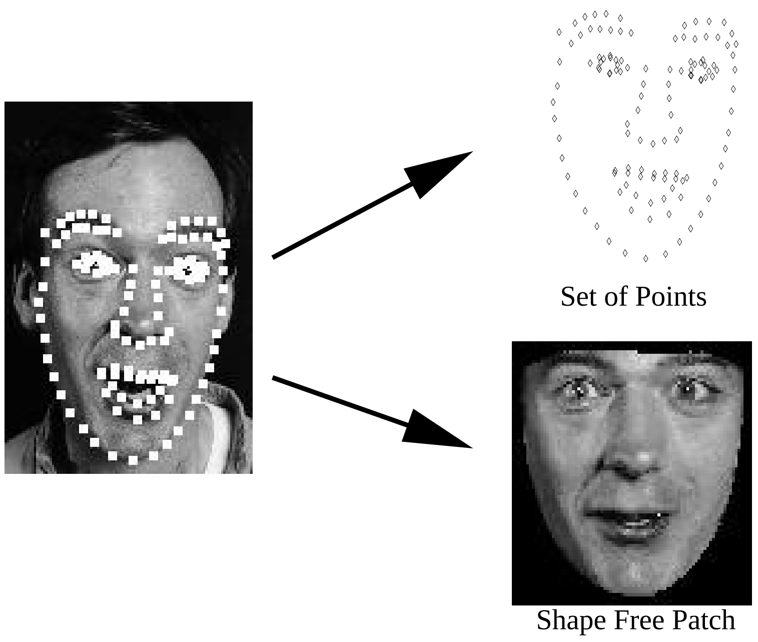


Figure 5.1: Each training example in split into a set of points and a 'shape-free' image patch

caused by the exaggerated expression - that is mostly taken account of by the shape.

To minimise the effect of global lighting variation, we normalise the example samples by applying a scaling, α, and offset, β,

$$\mathbf{g} = (\mathbf{g}_{im} - \beta\mathbf{1})/\alpha \tag{5.1}$$

The values of α and β are chosen to best match the vector to the normalised mean. Let $\bar{\mathbf{g}}$ be the mean of the normalised data, scaled and offset so that the sum of elements is zero and the variance of elements is unity. The values of α and β required to normalise $\mathbf{g}_{im}$ are then given by

$$\alpha = \mathbf{g}_{im}.\bar{\mathbf{g}} \quad , \quad \beta = (\mathbf{g}_{im}.\mathbf{1})/n \tag{5.2}$$

where n is the number of elements in the vectors.

Of course, obtaining the mean of the normalised data is then a recursive process, as the normalisation is defined in terms of the mean. A stable solution can be found by using one of the examples as the first estimate of the mean, aligning the others to it (using 5.1 and 5.2), re-estimating the mean and iterating.

By applying PCA to the normalised data we obtain a linear model:

$$\mathbf{g} = \bar{\mathbf{g}} + \mathbf{P}_g\mathbf{b}_g \tag{5.3}$$

where $\bar{\mathbf{g}}$ is the mean normalised grey-level vector, $\mathbf{P}_g$ is a set of orthogonal *modes of variation* and $\mathbf{b}_g$ is a set of grey-level parameters.

The texture in the image frame can be generated from the texture parameters, $\mathbf{b}_g$, and the normalisation parameters α, β. For linearity we represent these in a vector $\mathbf{u} = (\alpha - 1, \beta)^T$ (See Appendix E). In this form the identity transform is represented by the zero vector. The texture in the image frame is then given by

$$\mathbf{g}_{im} = T_{\mathbf{u}}(\bar{\mathbf{g}} + \mathbf{P}_g\mathbf{b}_g) = (1 + u_1)(\bar{\mathbf{g}} + \mathbf{P}_g\mathbf{b}_g) + u_2\mathbf{1} \tag{5.4}$$

5.2 Combined Appearance Models

The shape and texture of any example can thus be summarised by the parameter vectors $\mathbf{b}_s$ and $\mathbf{b}_g$. Since there may be correlations between the shape and texture variations, we apply a further PCA to the data as follows. For each example we generate the concatenated vector

$$\mathbf{b} = \begin{pmatrix} \mathbf{W}_s\mathbf{b}_s \\ \mathbf{b}_g \end{pmatrix} = \begin{pmatrix} \mathbf{W}_s\mathbf{P}_s^T(\mathbf{x} - \bar{\mathbf{x}}) \\ \mathbf{P}_g^T(\mathbf{g} - \bar{\mathbf{g}}) \end{pmatrix} \tag{5.5}$$

where $\mathbf{W}_s$ is a diagonal matrix of weights for each shape parameter, allowing for the difference in units between the shape and grey models (see below). We apply a PCA on these vectors, giving a further model

$$\mathbf{b} = \mathbf{Q}\mathbf{c} \tag{5.6}$$

where $\mathbf{Q}$ are the eigenvectors and $\mathbf{c}$ is a vector of *appearance* parameters controlling both the shape and grey-levels of the model. Since the shape and grey-model parameters have zero mean, $\mathbf{c}$ does too.

Note that the linear nature of the model allows us to express the shape and grey-levels directly as functions of $\mathbf{c}$

$$\mathbf{x} = \bar{\mathbf{x}} + \mathbf{P}_s\mathbf{W}_s^{-1}\mathbf{Q}_s\mathbf{c} \quad , \quad \mathbf{g} = \bar{\mathbf{g}} + \mathbf{P}_g\mathbf{Q}_g\mathbf{c} \tag{5.7}$$

where

$$\mathbf{Q} = \begin{pmatrix} \mathbf{Q}_s \\ \mathbf{Q}_g \end{pmatrix} \tag{5.8}$$

An example image can be synthesised for a given $\mathbf{c}$ by generating the shape-free grey-level image from the vector $\mathbf{g}$ and warping it using the control points described by $\mathbf{x}$.

5.2.1 Choice of Shape Parameter Weights

The elements of $\mathbf{b}_s$ have units of distance, those of $\mathbf{b}_g$ have units of intensity, so they cannot be compared directly. Because $\mathbf{P}_g$ has orthogonal columns, varying $\mathbf{b}_g$ by one unit moves $\mathbf{g}$ by one unit. To make $\mathbf{b}_s$ and $\mathbf{b}_g$ commensurate, we must estimate the effect of varying $\mathbf{b}_s$ on the sample $\mathbf{g}$. To do this we systematically displace each element of $\mathbf{b}_s$ from its optimum value on each training example, and sample the image given the displaced shape. The RMS change in $\mathbf{g}$ per unit change in shape parameter b_s gives the weight w_s to be applied to that parameter in equation (5.5).

A simpler alternative is to set $\mathbf{W}_s = r\mathbf{I}$ where r^2 is the ratio of the total intensity variation to the total shape variation (in the normalised frames). In practise the synthesis and search algorithms are relatively insensitive to the choice of $\mathbf{W}_s$.

5.3 Example: Facial Appearance Model

We used the method described above to build a model of facial appearance. We used a training set of 400 images of faces, each labelled with 122 points around the main features (Figure 5.1). From this we generated a shape model with 23 parameters, a shape-free grey model with 114 parameters and a combined appearance model with only 80 parameters required to explain 98% of the observed variation. The model uses about 10,000 pixel values to make up the face patch.

Figures 5.2 and 5.3 show the effects of varying the first two shape and grey-level model parameters through ± 3 standard deviations, as determined from the training set. The first parameter corresponds to the largest eigenvalue of the covariance matrix, which gives its variance across the training set. Figure 5.4 shows the effect of varying the first four appearance model parameters, showing changes in identity, pose and expression.

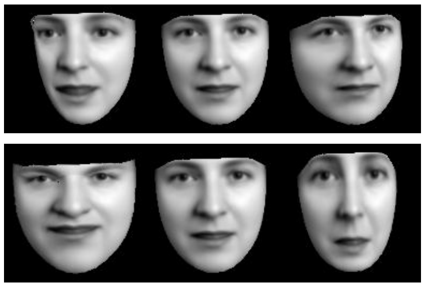


Figure 5.2: First two modes of shape variation (± 3 sd)

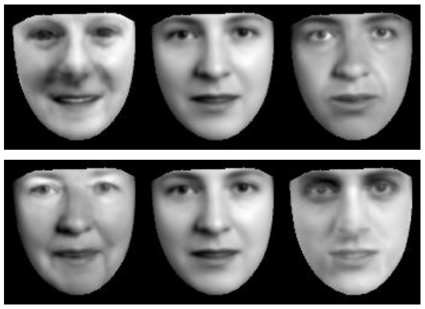


Figure 5.3: First two modes of grey-level variation (± 3 sd)

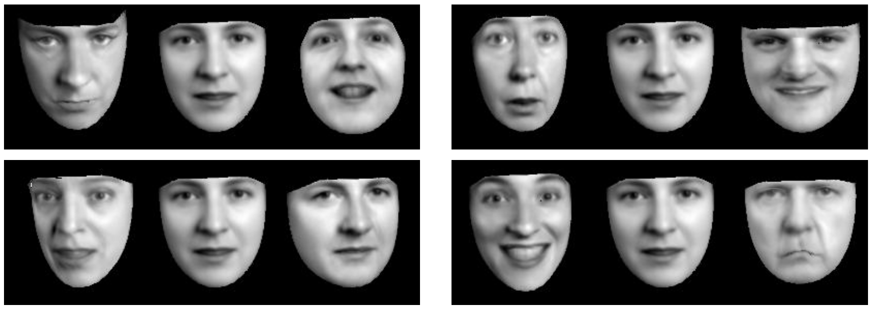


Figure 5.4: First four modes of appearance variation (± 3 sd)

5.4 Approximating a New Example

Given a new image, labelled with a set of landmarks, we can generate an approximation with the model. We follow the steps in the previous section to obtain $\mathbf{b}$, combining the shape and grey-level parameters which match the example. Since $\mathbf{Q}$ is orthogonal, the combined appearance model parameters, $\mathbf{c}$ are given by

$$\mathbf{c} = \mathbf{Q}^T\mathbf{b} \tag{5.9}$$

The full reconstruction is then given by applying equations (5.7), inverting the grey-level normalisation, applying the appropriate pose to the points and projecting the grey-level vector into the image.

For example, Figure 5.5 shows a previously unseen image alongside the model reconstruction of the face patch (overlaid on the original image).

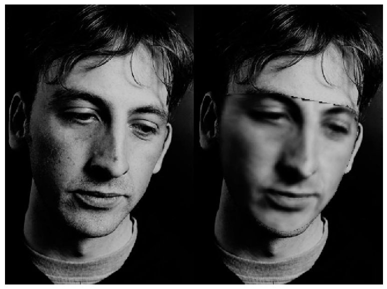


Figure 5.5: Example of combined model representation (right) of a previously unseen face image (left)

Chapter 6

Image Interpretation with Models

6.1 Overview

To interpret an image using a model, we must find the set of parameters which best match the model to the image. This set of parameters defines the shape, position and possibly appearance of the target object in an image, and can be used for further processing, such as to make measurements or to classify the object.

There are several approaches which could be taken to matching a model instance to an image, but all can be thought of as optimising a cost function. For a set of model parameters, **c**, we can generate an instance of the model projected into the image. We can compare this hypothesis with the target image, to get a fit function $F(\mathbf{c})$. The best set of parameters to interpret the object in the image is then the set which optimises this measure. For instance, if $F(\mathbf{c})$ is an error measure, which tends to zero for a perfect match, we would like to choose parameters, **c**, which minimise the error measure.

Thus, in theory all we have to do is to choose a suitable fit function, and use a general purpose optimiser to find the minimum. The minimum is defined only by the choice of function, the model and the image, and is independent of which optimisation method is used to find it. However, in practice, care must be taken to choose a function which can be optimised rapidly and robustly, and an optimisation method to match.

6.2 Choice of Fit Function

Ideally we would like to choose a fit function which represents the probability that the model parameters describe the target image object, $P(\mathbf{c}|\mathbf{I})$ (where **I** represents the image). We then choose the parameters which maximise this probability.

In the case of the shape models described above, the parameters we can vary are the shape parameters, **b**, and the pose parameters X_t, Y_t, s, θ. For the appearance models they are the appearance model parameters, **c** and the pose parameters.

The quality of fit of an appearance model can be assessed by measuring the difference between the target image and a synthetic image generated from the model. This is described in detail in Chapter 8.

The form of the fit measure for the shape models alone is harder to determine. If we assume that the shape model represents boundaries and strong edges of the object, a useful measure is the distance between a given model point and the nearest strong edge in the image (Figure 6.1).

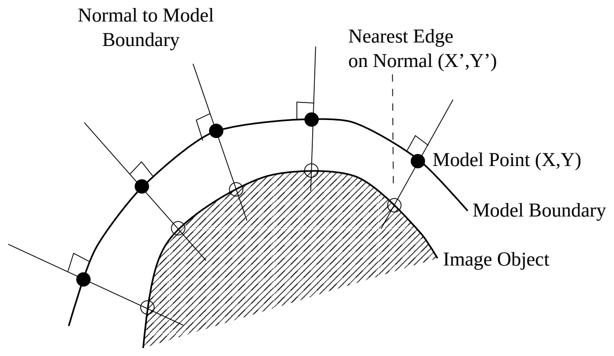


Figure 6.1: An error measure can be derived from the distance between model points and strongest nearby edges.

If the model point positions are given in the vector $\mathbf{X}$, and the nearest edge points to each model point are $\mathbf{X}'$, then an error measure is

$$F(\mathbf{b}, X_t, Y_t, s, \theta) = |\mathbf{X}' - \mathbf{X}|^2 \tag{6.1}$$

Alternatively, rather than looking for the best nearby edges, one can search for structure nearby which is most similar to that occuring at the given model point in the training images (see below).

It should be noted that this fit measure relies upon the target points, $\mathbf{X}'$, begin the correct points. If some are incorrect, due to clutter or failure of the edge/feature detectors, Equation (6.1) will not be a true measure of the quality of fit.

An alternative approach is to sample the image around the current model points, and determine how well the image samples match models derived from the training set. This approach was taken by Haslam *et al* [30].

6.3 Optimising the Model Fit

Given no initial knowledge of where the target object lies in an image, finding the parameters which optimise the fit is a a difficult general optimisation problem. This can be tackled with general global optimisation techniques, such as Genetic Algorithms or Simulated Annealing [32].

If, however, we have an initial approximation to the correct solution (we know roughly where the target object is in an image, due to prior processing), we can use local optimisation techniques such as Powell's method or Simplex. A good overview of practical numeric optimisation is given by Press *et al* [57].

However, we can take advantage of the form of the fit function to locate the optimum rapidly. We derive two algorithms which amounts to directed search of the parameter space - the Active

Shape Model and the Active Appearance Model.

In the following chapters we describe the Active Shape Model (which matches a shape model to an image) and the Active Appearance Model (which matches a full model of appearance to an image).

Chapter 7

Active Shape Models

7.1 Introduction

Given a rough starting approximation, an instance of a model can be fit to an image. By choosing a set of shape parameters, $\mathbf{b}$for the model we define the shape of the object in an object-centred co-ordinate frame. We can create an instance $\mathbf{X}$ of the model in the image frame by defining the position, orientation and scale, using Equation 4.11.

An iterative approach to improving the fit of the instance, $\mathbf{X}$, to an image proceeds as follows:

1. Examine a region of the image around each point $\mathbf{X}_i$ to find the best nearby match for the point $\mathbf{X}'_i$
2. Update the parameters $(X_t, Y_t, s, \theta, \mathbf{b})$ to best fit the new found points $\mathbf{X}$
3. Repeat until convergence.

In practise we look along profiles normal to the model boundary through each model point (Figure 7.1). If we expect the model boundary to correspond to an edge, we can simply locate the strongest edge (including orientation if known) along the profile. The position of this gives the new suggested location for the model point.

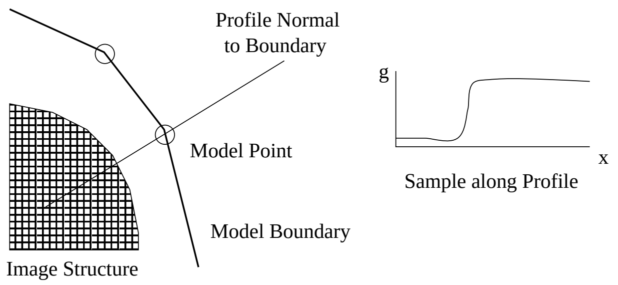


Figure 7.1: At each model point sample along a profile normal to the boundary

However, model points are not always placed on the strongest edge in the locality - they may represent a weaker secondary edge or some other image structure. The best approach is to learn from the training set what to look for in the target image. This is done by sampling along the profile normal to the boundary in the training set, and building a statistical model of the grey-level structure.

7.2 Modelling Local Structure

Suppose for a given point we sample along a profile k pixels either side of the model point in the i^{th} training image. We have $2k+1$ samples which can be put in a vector $\mathbf{g}_i$. To reduce the effects of global intensity changes we sample the derivative along the profile, rather than the absolute grey-level values. We then normalise the sample by dividing through by the sum of absolute element values,

$$\mathbf{g}_i \rightarrow \frac{1}{\sum_j |g_{ij}|}\mathbf{g}_i \tag{7.1}$$

We repeat this for each training image, to get a set of normalised samples $\{\mathbf{g}_i\}$ for the given model point. We assume that these are distributed as a multivariate gaussian, and estimate their mean $\bar{\mathbf{g}}$ and covariance $\mathbf{S}_g$. This gives a statistical model for the grey-level profile about the point. This is repeated for every model point, giving one grey-level model for each point.

The quality of fit of a new sample, $\mathbf{g}_s$, to the model is given by

$$f(\mathbf{g}_s) = (\mathbf{g}_s - \bar{\mathbf{g}})^T \mathbf{S}_g^{-1}(\mathbf{g}_s - \bar{\mathbf{g}}) \tag{7.2}$$

This is the Mahalanobis distance of the sample from the model mean, and is linearly related to the log of the probability that $\mathbf{g}_s$ is drawn from the distribution. Minimising $f(\mathbf{g}_s)$ is equivalent to maximising the probability that $\mathbf{g}_s$ comes from the distribution.

During search we sample a profile m pixels either side of the current point ($m > k$). We then test the quality of fit of the corresponding grey-level model at each of the $2(m-k)+1$ possible positions along the sample (Figure 7.2) and choose the one which gives the best match (lowest value of $f(\mathbf{g}_s)$).

This is repeated for every model point, giving a suggested new position for each point. We then apply one iteration of the algorithm given in (4.8) to update the current pose and shape parameters to best match the model to the new points.

7.3 Multi-Resolution Active Shape Models

To improve the efficiency and robustness of the algorithm, it is implement in a multi-resolution framework. This involves first searching for the object in a coarse image, then refining the location in a series of finer resolution images. This leads to a faster algorithm, and one which is less likely to get stuck on the wrong image structure.

For each training and test image, a gaussian image pyramid is built [9]. The base image (level 0) is the original image. The next image (level 1) is formed by smoothing the original then

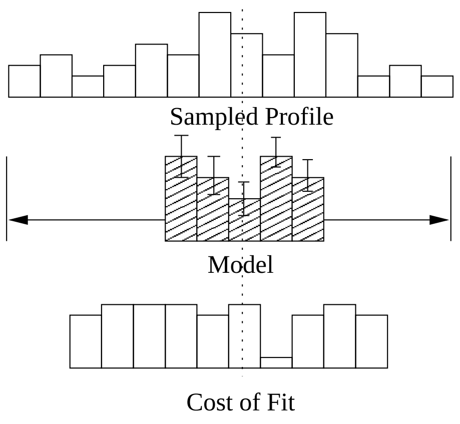


Figure 7.2: Search along sampled profile to find best fit of grey-level model

subsampling to obtain an image with half the number of pixels in each dimension. Subsequent levels are formed by further smoothing and sub-sampling (Figure 7.3).

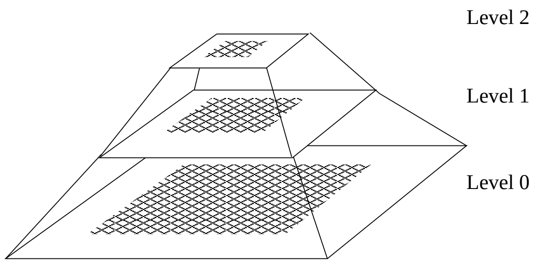


Figure 7.3: A gaussian image pyramid is formed by repeated smoothing and sub-sampling

During training we build statistical models of the grey-levels along normal profiles through each point, at each level of the gaussian pyramid. We usually use the same number of pixels in each profile model, regardless of level. Since the pixels at level L are 2^L times the size of those of the original image, the models at the coarser levels represent more of the image (Figure 7.4). Similarly, during search we need only search a few pixels, (n_s), either side of the current point position at each level. At coarse levels this will allow quite large movements, and the model should converge to a good solution. At the finer resolution we need only modify this solution by small amounts.

When searching at a given resolution level, we need a method of determining when to change to a finer resolution, or to stop the search. This is done by recording the number of times that the best found pixel along a search profile is within the central 50% of the profile (*ie* the best point is within $n_s/2$ pixels of the current point). When a sufficient number (eg $\geq 90\%$) of the points are so found, the algorithm is declared to have converged at that resolution. The current model is projected into the next image and run to convergence again. When convergence is reached on the finest resolution, the search is stopped.

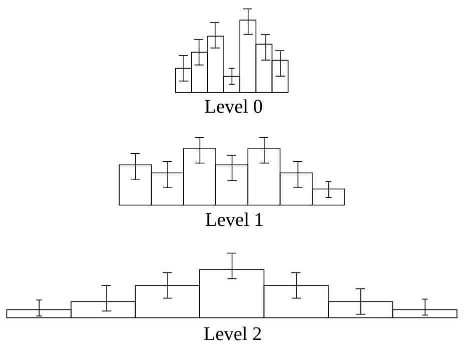


Figure 7.4: Statistical models of grey-level profiles represent the same number of pixels at each level

To summarise, the full MRASM search algorithm is as follows:

1. Set $L = L_{max}$

2. While $L \geq 0$

   (a) Compute model point positions in image at level L.

   (b) Search at n_s points on profile either side each current point

   (c) Update pose and shape parameters to fit model to new points

   (d) Return to (2a) unless more than p_{close} of the points are found close to the current position, or N_{max} iterations have been applied at this resolution.

   (e) If $L > 0$ then $L \rightarrow (L-1)$

3. Final result is given by the parameters after convergence at level 0.

The model building process only requires the choice of three parameters:

Model Parameters	
n	Number of model points
t	Number of modes to use
k	Number of pixels either side of point to represent in grey-model

The number of points is dependent on the complexity of the object, and the accuracy with which one wishes to represent any boundaries. The number of modes should be chosen that a sufficient amount of object variation can be captured (see 4.9 above). The number of pixels to model in each pixel will depend on the width of the boundary structure (however, using 3 pixels either side has given good results for many applications).

The search algorithm has four parameters:

Search Parameters (Suggested default)	
L_{max}	Coarsest level of gaussian pyramid to search
n_s	Number of sample points either side of current point (2)
N_{max}	Maximum number of iterations allowed at each level (5)
p_{close}	Proportion of points found within $n_s/2$ of current pos. (0.9)

The levels of the gaussian pyramid to seach will depend on the size of the object in the image.

7.4 Examples of Search

Figure 7.5 demonstrates using the ASM to locate the features of a face. The model instance is placed near the centre of the image and a coarse to fine search performed. The search starts at level 3 (1/8 the resolution in x and y compared to the original image). Large movements are made in the first few iterations, getting the position and scale roughly correct. As the search progresses to finer resolutions more subtle adjustments are made. The final convergence (after a total of 18 iterations) gives a good match to the target image. In this case at most 5 iterations were allowed at each resolution, and the algorithm converges in much less than a second (on a 200MHz PC).

Figure 7.6 demonstrates how the ASM can fail if the starting position is too far from the target. Since it is only searching along profiles around the current position, it cannot correct for large displacements from the correct position. It will either diverge to infinity, or converge to an incorrect solution, doing its best to match the local image data. In the case shown it has been able to locate half the face, but the other side is too far away.

Figure 7.7 demonstrates using the ASM of the cartilage to locate the structure in a new image. In this case the search starts at level 2, samples at 2 points either side of the current point and allows at most 5 iterations per level. A detailed description of the application of such a model is given by Solloway *et. al.* in [64].

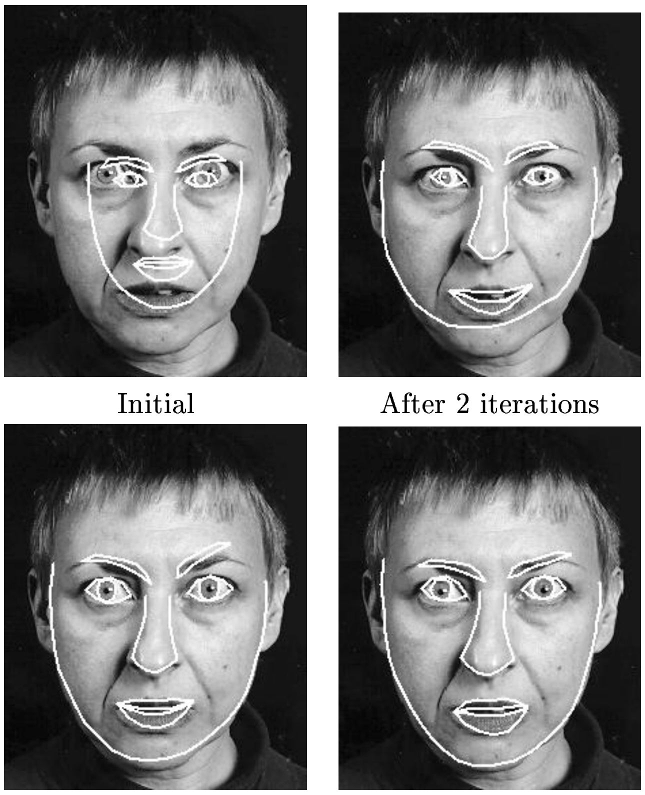


Figure 7.5: Search using Active Shape Model of a face

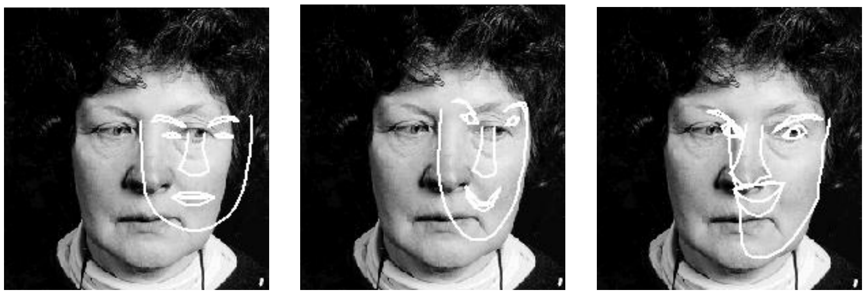


Figure 7.6: Search using Active Shape Model of a face, given a poor starting point. The ASM is a local method, and may fail to locate an acceptable result if initialised too far from the target

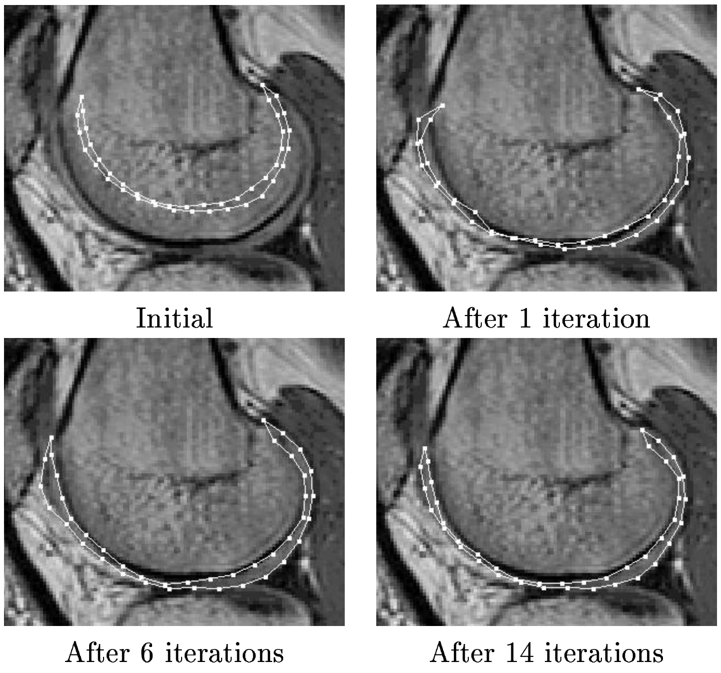


Figure 7.7: Search using ASM of cartilage on an MR image of the knee

Chapter 8

Active Appearance Models

8.1 Introduction

The Active Shape Model search algorithm allowed us to locate points on a new image, making use of constraints of the shape models. One disadvantage is that it only uses shape constraints (together with some information about the image structure near the landmarks), and does not take advantage of all the available information - the texture across the target object. This can be modelled using an Appearance Model 5. In this chapter we describe an algorithm which allows us to find the parameters of such a model which generates a synthetic image as close as possible to a particular target image, assuming a reasonable starting approximation.

8.2 Background: Interpretation by Synthesis

In recent years many model-based approaches to the interpretation of images of deformable objects have been described. One motivation is to achieve robust performance by using the model to constrain solutions to be valid examples of the object modelled. A model also provides the basis for a broad range of applications by 'explaining' the appearance of a given image in terms of a compact set of model parameters. These parameters are useful for higher level interpretation of the scene. For instance, when analysing face images they may be used to characterise the identity, pose or expression of a face. In order to interpret a new image, an efficient method of finding the best match between image and model is required.

Various approaches to modelling variability have been described. The most common general approach is to allow a prototype to vary according to some physical model. Bajcsy and Kovacic [1] describe a volume model (of the brain) that also deforms elastically to generate new examples. Christensen *et al* [11] describe a viscous flow model of deformation which they also apply to the brain, but is very computationally expensive. Turk and Pentland [70] use principal component analysis to describe face images in terms of a set of basis functions, or 'eigenfaces'. Though valid modes of variation are learnt from a training set, and are more likely to be more appropriate than a 'physical' model, the eigenface is not robust to shape changes, and does not deal well with variability in pose and expression. However, the model can be matched to an image easily using correlation based methods.

Poggio and co-workers [24] [38] synthesise new views of an object from a set of example views. They fit the model to an unseen view by a stochastic optimisation procedure. This is slow, but can be robust because of the quality of the synthesised images. Cootes *et al* [14] describe a 3D model of the grey-level surface, allowing full synthesis of shape and appearance. However, they do not suggest a plausible search algorithm to match the model to a new image. Nastar *at al* [54] describe a related model of the 3D grey-level surface, combining physical and statistical modes of variation. Though they describe a search algorithm, it requires a very good initialisation. Lades *at al* [46] model shape and some grey level information using Gabor jets. However, they do not impose strong shape constraints and cannot easily synthesise a new instance.

Cootes *et al* [17] model shape and local grey-level appearance, using Active Shape Models (ASMs) to locate flexible objects in new images. Lanitis *at al* [47] use this approach to interpret face images. Having found the shape using an ASM, the face is warped into a normalised frame, in which a model of the intensities of the shape-free face is used to interpret the image. Edwards *at al* [23] extend this work to produce a combined model of shape and grey-level appearance, but again rely on the ASM to locate faces in new images. Our new approach can be seen as a further extension of this idea, using all the information in the combined appearance model to fit to the image.

In developing our new approach we have benefited from insights provided by two earlier papers. Covell [19] demonstrated that the parameters of an eigen-feature model can be used to drive shape model points to the correct place. The AAM described here is an extension of this idea. Black and Yacoob [4] use local, hand crafted models of image flow to track facial features, but do not attempt to model the whole face. The AAM can be thought of as a generalisation of this, in which the image difference patterns corresponding to changes in each model parameter are learnt and used to modify a model estimate.

In a parallel development Sclaroff and Isidoro have demonstrated 'Active Blobs' for tracking [61]. The approach is broadly similar in that they use image differences to drive tracking, learning the relationship between image error and parameter offset in an off-line processing stage. The main difference is that Active Blobs are derived from a single example, whereas Active Appearance Models use a training set of examples. The former use a single example as the original model template, allowing deformations consistent with low energy mesh deformations (derived using a Finite Element method). A simply polynomial model is used to allow changes in intensity across the object. AAMs learn what are valid shape and intensity variations from their training set.

Sclaroff and Isidoro suggest applying a robust kernel to the image differences, an idea we will use in later work. Also, since annotating the training set is the most time consuming part of building an AAM, the Active Blob approach may be useful for 'bootstrapping' from the first example.

8.3 Overview of AAM Search

We wish to treat interpretation as an optimisation problem in which we minimise the difference between a new image and one synthesised by the appearance model. A difference vector $\delta\mathbf{I}$ can be defined:

$$\delta \mathbf{I} = \mathbf{I_i} - \mathbf{I_m} \tag{8.1}$$

where $\mathbf{I_i}$ is the vector of grey-level values in the image, and $\mathbf{I_m}$, is the vector of grey-level values for the current model parameters.

To locate the best match between model and image, we wish to minimise the magnitude of the difference vector, $\Delta = |\delta \mathbf{I}|^2$, by varying the model parameters, $\mathbf{c}$. Since the appearance models can have many parameters, this appears at first to be a difficult high-dimensional optimisation problem. We note, however, that each attempt to match the model to a new image is actually a similar optimisation problem. We propose to learn something about how to solve this class of problems in advance. By providing a-priori knowledge of how to adjust the model parameters during during image search, we arrive at an efficient run-time algorithm. In particular, the spatial pattern in $\delta \mathbf{I}$, encodes information about how the model parameters should be changed in order to achieve a better fit. In adopting this approach there are two parts to the problem: learning the relationship between $\delta \mathbf{I}$ and the error in the model parameters, $\delta \mathbf{c}$ and using this knowledge in an iterative algorithm for minimising Δ.

8.4 Learning to Correct Model Parameters

The simplest model we could choose for the relationship between $\delta \mathbf{I}$ and the error in the model parameters (and thus the correction which needs to be made) is linear:

$$\delta \mathbf{c} = \mathbf{A} \delta \mathbf{I} \tag{8.2}$$

This turns out to be a good enough approximation to achieve acceptable results. To find $\mathbf{A}$, we perform multiple multivariate linear regression on a sample of known model displacements, $\delta \mathbf{c}$, and the corresponding difference images, $\delta \mathbf{I}$. We can generate these sets of random displacements by perturbing the 'true' model parameters for the images in which they are known. These can either be the original training images or synthetic images generated with the appearance model. In the latter case we know the parameters exactly, and the images are not corrupted by noise.

As well as perturbations in the model parameters, we also model small displacements in 2D position, scale, and orientation, and the intensity scaling and offset parameters. These six extra parameters are included in the regression. To retain linearity we represent small changes in pose using $\delta \mathbf{t} = (\delta s_x, \delta s_y, \delta t_x, \delta t_y)^T$ (See Appendix D). Similarly we represent small changes in the texture using $\delta \mathbf{u} = (\delta u_1, \delta u_2)^T$ (Appendix E). In order to obtain a well-behaved relationship it is important to choose carefully the frame of reference in which the image difference is calculated. The most suitable frame of reference is the shape-normalised patch described in Chapter 5.

Let $\mathbf{c}_0$ be the known appearance model parameters for the current image, $\mathbf{t}_0$ be the optimal pose parameters and $\mathbf{u}_0$ the optimal texture transformation.

We calculate a difference thus:

- Randomly generate model, pose and texture transform parameter displacements, $\delta \mathbf{c}$, $\delta \mathbf{t}$, $\delta \mathbf{u}$
- Let $\mathbf{c} = \delta \mathbf{c} + \mathbf{c}_0$

- Let $\mathbf{t}$ be the pose parameters such that $T_{\mathbf{t}}(\mathbf{x}) = T_{\mathbf{t}_0}(T_{\delta\mathbf{t}}(\mathbf{x}))$ (See Appendix D).
- Let $\mathbf{u}$ be the texture transform parameters such that $T_{\mathbf{u}}(\mathbf{g}) = T_{\mathbf{u}_0}(T_{\delta\mathbf{u}}(\mathbf{x}))$ (See Appendix E).
- Given $\mathbf{c}$ and $\mathbf{t}$ generate the shape in the image frame, $\mathbf{X}$, and the normalised model texture vector, $\mathbf{g}_m$.
- Given the shape, $\mathbf{X}$, sample from the image to obtain $\mathbf{g}_{im}$.
- Project the sample into the texture model frame using $\mathbf{g}_s = T_{\mathbf{u}}^{-1}(\mathbf{g}_s)$
- The sample error is then $\delta\mathbf{g} = \mathbf{g}_s - \mathbf{g}_m$.

The training algorithm is then simply to randomly displace the model, pose and texture transformation parameters from the best fit to each training image, recording the displacements $\delta\mathbf{c}$, $\delta\mathbf{t}$, $\delta\mathbf{u}$ and the sampled error $\delta\mathbf{g}$.

We then perform multi-variate regression to obtain the relationships

$$\begin{aligned}\delta\mathbf{c} &= \mathbf{R}_c\delta\mathbf{g}\\ \delta\mathbf{t} &= \mathbf{R}_t\delta\mathbf{g}\\ \delta\mathbf{u} &= \mathbf{R}_u\delta\mathbf{g}\end{aligned} \tag{8.3}$$

(See Appendix F).

The best range of values of $\delta\mathbf{c}$, $\delta\mathbf{t}$ and $\delta\mathbf{u}$ to use during training is determined experimentally. Ideally we seek to model a relationship that holds over as large a range errors, $\delta\mathbf{g}$, as possible. However, the real relationship is found to be linear only over a limited range of values. Our experiments on the face model suggest that the optimum perturbation was around 0.5 standard deviations (over the training set) for each model parameter, about 10% in scale, the equivalent of 3 pixels translation and about 10% in texture scaling.

8.4.1 Results For The Face Model

We applied the above algorithm to the face model described in section 5.3. After performing linear regression, we can calculate an R^2 statistic for each parameter perturbation, δc_i to measure how well the displacement is 'predicted' by the error vector $\delta\mathbf{g}$. The average R^2 value for the 80 parameters was 0.82, with a maximum of 0.98 (the 1st parameter) and a minimum of 0.48.

We can visualise the effects of the perturbation as follows. If $\mathbf{a}_i$ is the i^{th} row of the regression matrix $\mathbf{A}$, the predicted change in the i^{th} parameter, δc_i is given by

$$\delta c_i = \mathbf{a}_i.\delta\mathbf{g} \tag{8.4}$$

and $\mathbf{a}_i$ gives the weight attached to different areas of the sampled patch when estimating the displacement. Figure 8.1 shows the weights corresponding to changes in the pose parameters, (s_x, s_y, t_x, t_y). Bright areas are positive weights, dark areas negative. As one would expect, the x and y displacement weights are similar to x and y derivative images. Similar results are obtained for weights corresponding to the appearance model parameters

Figure 8.2 and 8.3 show the first and third modes and corresponding displacement weights. The areas which exhibit the largest variations for the mode are assigned the largest weights by the training process.

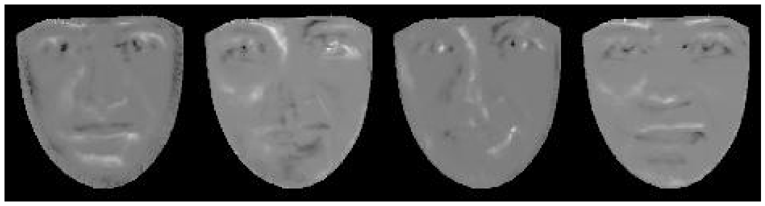


Figure 8.1: Weights corresponding to changes in the pose parameters, (s_x, s_y, t_x, t_y)

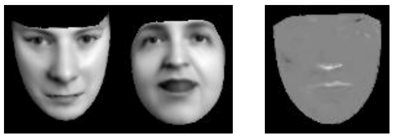


Figure 8.2: First mode and displacement weights

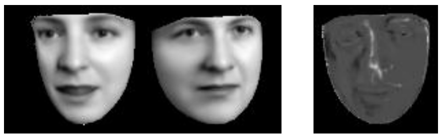


Figure 8.3: Third mode and displacement weights

8.4.2 Perturbing The Face Model

To examine the performance of the prediction, we systematically displaced the face model from the true position on a set of 10 test images, and used the model to predict the displacement given the sampled error vector. Figures 8.4 and 8.5 show the predicted translations against the actual translations. There is a good linear relationship within about 4 pixels of zero. Although this breaks down with larger displacements, as long as the prediction has the same sign as the actual error, and does not over-predict too far, an iterative updating scheme should converge. In this case up to 20 pixel displacements in x and about 10 in y should be correctable.

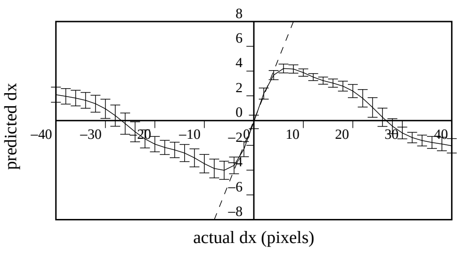


Figure 8.4: Predicted dx vs actual dx. Errorbars are 1 standard error

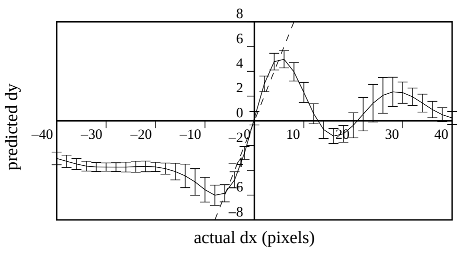


Figure 8.5: Predicted dy vs actual dy. Errorbars are 1 standard error

We can, however, extend this range by building a multi-resolution model of object appearance. We generate Gaussian pyramids for each of our training images, and generate an appearance model for each level of the pyramid. Figure 8.6 shows the predictions of models displaced in x at three resolutions. L0 is the base model, with about 10,000 pixels. L1 has about 2,500 pixels and L2 about 600 pixels.

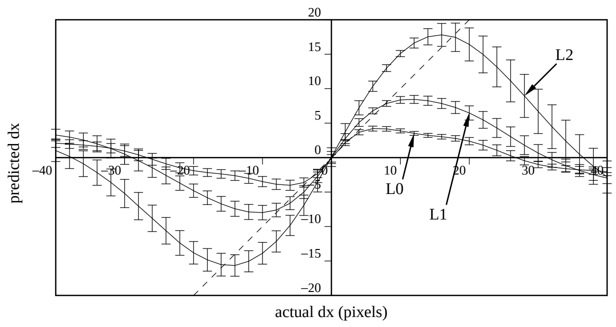


Figure 8.6: Predicted dx vs actual dx for 3 levels of a Multi-Resolution model. L0: 10000 pixels, L1: 2500 pixels, L2: 600 pixels. Errorbars are 1 standard error

The linear region of the curve extends over a larger range at the coarser resolutions, but is less accurate than at the finest resolution. Similar results are obtained for variations in other pose parameters and the model parameters.

Figure 8.7 shows the predicted displacements of s_x and s_y against the actual displacements. Figure 8.8 shows the predicted displacements of the first two model parameters c_1 and c_2 (in units of standard deviations) against the actual. In all cases there is a central linear region, suggesting an iterative algorithm will converge when close enough to the solution.

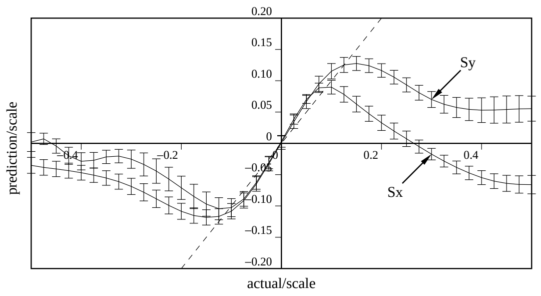


Figure 8.7: Predicted s_x and s_y vs actual

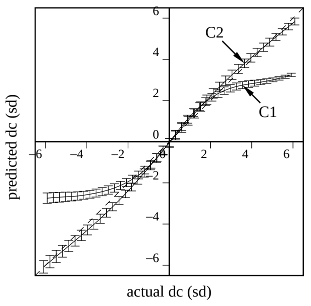


Figure 8.8: Predicted c_1 and c_2 vs actual

8.5 Iterative Model Refinement

Given a method for predicting the correction which needs to made in the model parameters we can construct an iterative method for solving our optimisation problem.

Given the current estimate of model parameters, $\mathbf{c}_0$, and the normalised image sample at the current estimate, $\mathbf{g}_s$, one step of the iterative procedure is as follows:

- Evaluate the error vector $\delta\mathbf{g}_0 = \mathbf{g}_s - \mathbf{g}_m$

- Evaluate the current error $E_0 = |\delta \mathbf{g}_0|^2$
- Compute the predicted displacement, $\delta \mathbf{c} = \mathbf{A} \delta \mathbf{g}_0$
- Set $k = 1$
- Let $\mathbf{c}_1 = \mathbf{c}_0 - k\delta \mathbf{c}$
- Sample the image at this new prediction, and calculate a new error vector, $\delta \mathbf{g}_1$
- If $|\delta \mathbf{g}_1|^2 < E_0$ then accept the new estimate, $\mathbf{c}_1$,
- Otherwise try at $k = 1.5$, $k = 0.5$, $k = 0.25$ etc.

This procedure is repeated until no improvement is made to the error, $|\delta \mathbf{g}|^2$, and convergence is declared.

We use a multi-resolution implementation, in which we iterate to convergence at each level before projecting the current solution to the next level of the model. This is more efficient and can converge to the correct solution from further away than search at a single resolution.

8.5.1 Examples of Active Appearance Model Search

We used the face AAM to search for faces in previously unseen images. Figure 8.9 shows the best fit of the model given the image points marked by hand for three faces. Figure 8.10 shows frames from a AAM search for each face, each starting with the mean model displaced from the true face centre.

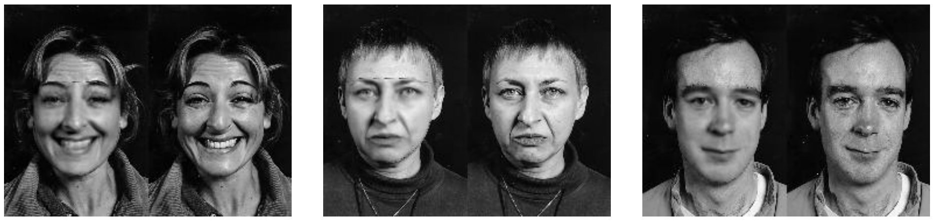


Figure 8.9: Reconstruction (left) and original (right) given original landmark points

As an example of applying the method to medical images, we built an Appearance Model of part of the knee as seen in a slice through an MR image. The model was trained on 30 examples, each labelled with 42 landmark points. Figure 8.11 shows the effect of varying the first two appearance model parameters. Figure 8.12 shows the best fit of the model to a new image, given hand marked landmark points. Figure 8.13 shows frames from an AAM search from a displaced position.

8.6 Experimental Results

To obtain a quantitative evaluation of the performance of the algorithm we trained a model on 88 hand labelled face images, and tested it on a different set of 100 labelled images. Each face was about 200 pixels wide.

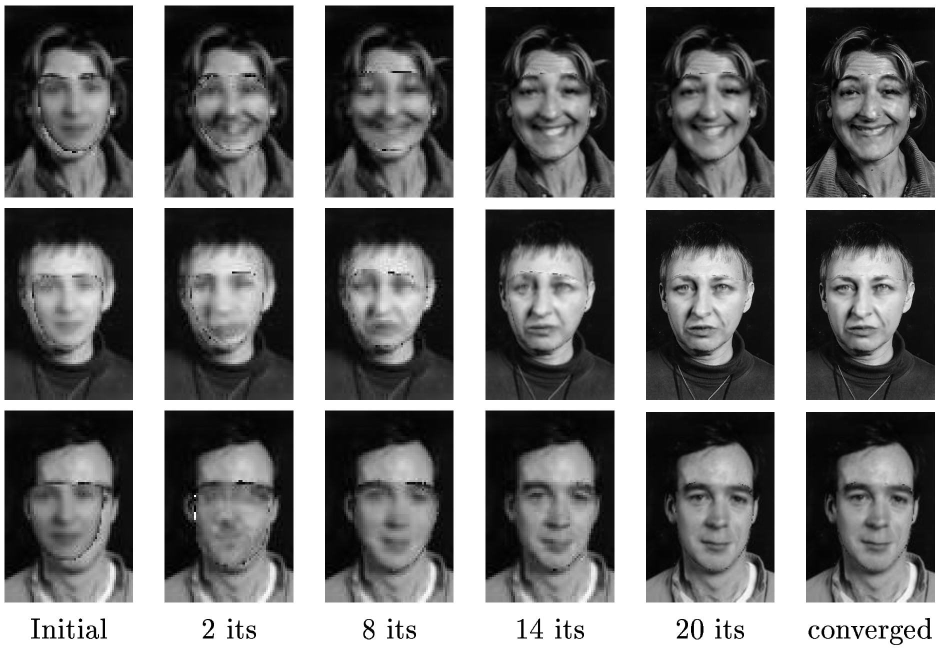


Figure 8.10: Multi-Resolution search from displaced position

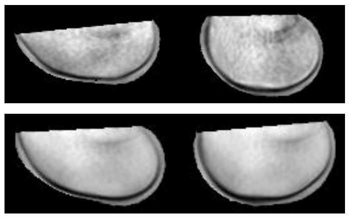


Figure 8.11: First two modes of appearance variation of knee model

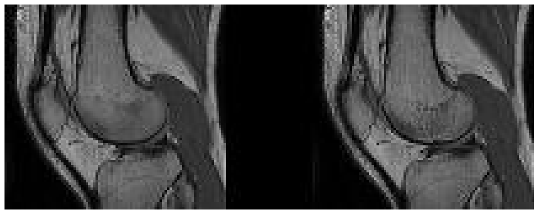


Figure 8.12: Best fit of knee model to new image given landmarks

On each test image we systematically displaced the model from the true position by ± 15 pixels in x and y, and changed its scale by $\pm 10\%$. We then ran the multi-resolution search, starting with the mean appearance model. 2700 searches were run in total, each taking an average of 4.1 seconds on a Sun Ultra. Of those 2700, 519 (19%) failed to converge to a satisfactory result (the mean point position error was greater than 7.5 pixels per point). Of those that did converge, the RMS error between the model centre and the target centre was $(0.8, 1.8)$ pixels. The s.d. of the model scale error was 6%. The mean magnitude of the final image error vector in the normalised frame relative to that of the best model fit given the marked points, was 0.88 (sd: 0.1), suggesting that the algorithm is locating a better result than that provided by the marked points. Because it is explicitly minimising the error vector, it will compromise the shape if that leads to an overall improvement of the grey-level fit.

Figure 8.14 shows the mean intensity error per pixel (for an image using 256 grey-levels) against the number of iterations, averaged over a set of searches at a single resolution. In

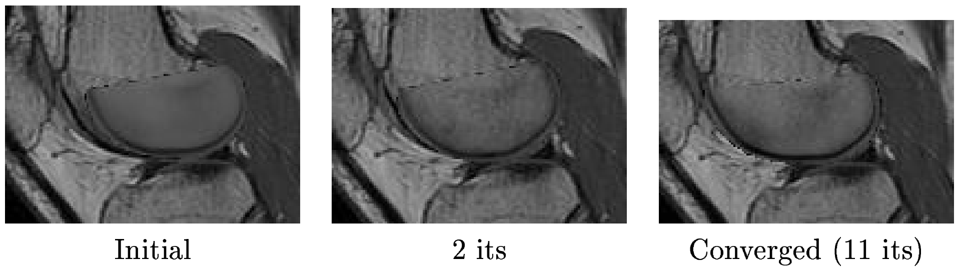


Figure 8.13: Multi-Resolution search for knee

each case the model was initially displaced by up to 15 pixels. The dotted line gives the mean reconstruction error using the hand marked landmark points, suggesting a good result is obtained by the search.

Figure 8.15 shows the proportion of 100 multi-resolution searches which converged correctly given starting positions displaced from the true position by up to 50 pixels in x and y. The model displays good results with up to 20 pixels (10% of the face width) displacement.

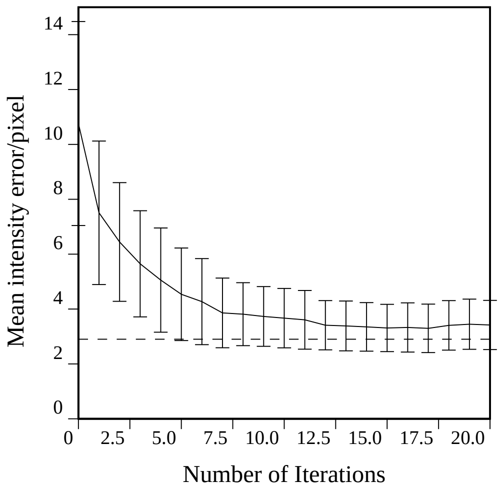


Figure 8.14: Mean intensity error as search progresses. Dotted line is the mean error of the best fit to the landmarks.

To be continued ...

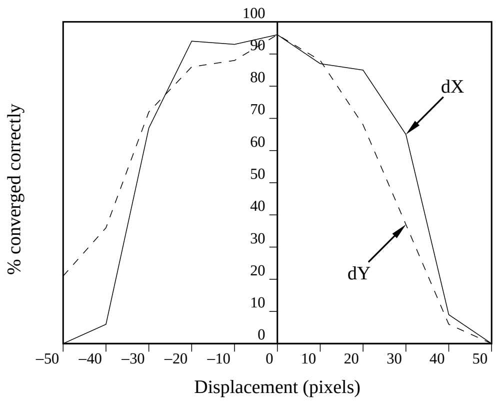


Figure 8.15: Proportion of searches which converged from different initial displacements

Chapter 9

Variations on the Basic AAM

In this chapter we describe modifications to the basic AAM algorithm aimed at improving the speed and robustness of search. Since some regions of the model may change little when parameters are varied, we need only sample the image in regions where significant changes are expected. This should reduce the cost of each iteration.

The original formulation manipulates the combined shape and grey-level parameters directly. An alternative approach is to use image residuals to drive the shape parameters, computing the grey-level parameters directly from the image given the current shape. This approach may be useful when there are few shape modes and many grey-level modes.

9.1 Sub-sampling During Search

In the original formulation, during search we sample all the points in the model to obtain $\mathbf{g}_s$, with which we predict the change to the model parameters. There may be 10000 or more such pixels, but fewer than 100 parameters. There is thus considerable redundancy, and it may be possible to obtain good results by sampling at only a sub-set of the modelled pixels. This could significantly reduce the computational cost of the algorithm.

The change in the i^{th} parameter, δc_i, is given by

$$\delta c_i = \mathbf{A}_i \delta \mathbf{g} \tag{9.1}$$

Where $\mathbf{A}_i$ is the i^{th} row of $\mathbf{A}$.

The elements of $\mathbf{A}_i$ indicate the significance of the corresponding pixel in the calculation of the change in the parameter. To choose the most useful subset for a given parameter, we simply sort the elements by absolute value and select the largest. However, the pixels which best predict changes to one parameter may not be useful for any other parameter.

To select a useful subset for all parameters we compute the best $u\%$ of elements for each parameter, then generate the union of such sets. If u is small enough, the union will be less than all the elements.

Given such a subset, we perform a new multi-variate regression, to compute the relationship, $\mathbf{A}'$ between the changes in the subset of samples, $\delta \mathbf{g}'$, and the changes in parameters

$$\delta \mathbf{c} = \mathbf{A}' \delta \mathbf{g}' \tag{9.2}$$

Search can proceed as described above, but using only a subset of all the pixels.

9.2 Search Using Shape Parameters

The original formulation manipulates the parameters, $\mathbf{c}$. An alternative approach is to use image residuals to drive the shape parameters, $\mathbf{b}_s$, computing the grey-level parameters, $\mathbf{b}_g$, and thus $\mathbf{c}$, directly from the image given the current shape. This approach may be useful when there are few shape modes and many grey-level modes.

The update equation in this case has the form

$$\delta \mathbf{b}_s = \mathbf{B} \delta \mathbf{g} \tag{9.3}$$

where in this case $\delta \mathbf{g}$ is given by the difference between the current image sample $\mathbf{g}_s$ and the best fit of the grey-level model to it, $\mathbf{g}_m$,

$$\begin{aligned} \delta \mathbf{g} &= \mathbf{g}_s - \mathbf{g}_m \\ &= \mathbf{g}_s - (\bar{\mathbf{g}} + \mathbf{P}_g \mathbf{b}_g) \end{aligned} \tag{9.4}$$

where $\mathbf{b}_g = \mathbf{P}_g^T (\mathbf{g}_s - \bar{\mathbf{g}})$.

During a training phase we use regression to learn the relationship, $\mathbf{B}$, between $\delta \mathbf{b}_s$ and $\delta \mathbf{g}$ (as given in (9.4)). Since any $\delta \mathbf{g}$ is orthogonal to the columns of $\mathbf{P}_g$, the update equation simplifies to

$$\begin{aligned} \delta \mathbf{b}_s &= \mathbf{B}(\mathbf{g}_s - \bar{\mathbf{g}}) \\ &= \mathbf{B}\mathbf{g}_s - \mathbf{b}_{offset} \end{aligned} \tag{9.5}$$

Thus one approach to fitting a model to an image is simply to keep track of the pose and shape parameters, $\mathbf{b}_s$. The grey-level parameters can be computed directly from the sample at the current shape. The constraints of the combined appearance model can be applied by computing $\mathbf{c}$ using (5.5), applying constraints then recomputing the shape parameters. As in the original formulation, the magnitude of the residual $|\delta \mathbf{g}|$ can be used to test for convergence.

In cases where there are significantly fewer modes of shape variation than combined appearance modes, this approach may be faster. However, since it is only indirectly driving the parameters controlling the full appearance, $\mathbf{c}$, it may not perform as well as the original formulation.

Note that we could test for convergence by monitoring changes in the shape parameters, or simply apply a fixed number of iterations at each resolution. In this case we do not need to use the grey-level model at all during search. We would just do a single match to the grey-levels sampled from the final shape. This may give a significantly faster algorithm.

9.3 Results of Experiments

To compare the variations on the algorithm described above, an appearance model was trained on a set of 300 labelled faces. This set contains several images of each of 40 people, with a variety of different expressions. Each image was hand annotated with 122 landmark points on the key features. From this data was built a shape model with 36 parameters, a grey-level model of 10000 pixels with 223 parameters and a combined appearance model with 93 parameters.

Three versions of the AAM were trained for these models. One with the original formulation, a second using a sub-set of 25% of the pixels to drive the parameters $\mathbf{c}$, and a third trained to drive the shape parameters, $\mathbf{b}_s$, alone.

A test set of 100 unseen new images (of the same set of people but with different expressions) was used to compare the performance of the algorithms. On each image the optimal pose was found from hand annotated landmarks. The model was displaced by (+15, 0 ,-15) pixels in x and y, the remaining parameters were set to zero and a multi-resolution search performed (9 tests per image, 900 in all).

Two search regimes were used. In the first a maximum of 5 iterations were allowed at each resolution level. Each iteration tested the model at $\mathbf{c} \rightarrow \mathbf{c} - k\delta\mathbf{c}$ for $k = 1.0, 0.5^1, \ldots, 0.5^4$, accepting the first that gave an improved result or declaring convergence if none did.

The second regime forced the update $\mathbf{c} \rightarrow \mathbf{c} - \delta\mathbf{c}$ without testing whether it was better or not, applying 5 steps at each resolution level.

The quality of fit was recorded in two ways;

- The RMS grey-level error per pixel in the normalised frame, $\sqrt{|\delta\mathbf{v}|^2/n_{pixels}}$
- The mean distance error per model point

For example, the result of the first search shown in Figure 8.10 above gives an RMS grey error of 0.46 per pixel and a mean distance error of 3.7 pixels.

Some searches will fail to converge to near the correct result. This is detected by a threshold on the mean distance error per model point. Those that have a mean error of > 7.5 pixels were considered to have failed to converge.

Table 9.1 summarises the results. The final errors recorded were averaged over those searches which converged successfully.. The top row corresponds to the original formulation of the AAM. It was the slowest, but on average gave the fewest failures and the smallest grey-level error. Forcing the iterations decreased the quality of the results, but was about 25% faster.

Sub-sampling considerably speeded up the search (taking only 30% of the time for full sampling) but was much less likely to converge correctly, and gave a poorer overall result.

Driving the shape parameters during search was faster still, but again lead to more failures than the original AAM. However, it did lead to more accurate location of the target points when the search converged correctly. This was at the expense of increasing the error in the grey-level match.

The best fit of the Appearance Model to the images given the labels gave a mean RMS grey error of 0.37 per pixel over the test set, suggesting the AAM was getting close to the best possible result most of the time.

Table 9.2 shows the results of a similar experiment in which the models were started from the best estimate of the correct pose, but with other model parameters initialised to zero. This shows a much reduced failure rate, but confirms the conclusions drawn from the first experiment. The search could fail even given the correct initial pose because some of the images contain quite exaggerated expressions and head movements, a long way from the mean. These were difficult to match to, even under the best conditions.

Driven Params	Sub-sample	Iterations		Failure Rate	Final Errors		Mean Time (ms)
		Max.	Forced		Point ± 0.05	Grey ± 0.005	
$\mathbf{c}$	100%	5	1	4.1%	4.2	0.45	3270
$\mathbf{c}$	100%	5	5	4.6%	4.4	0.46	2490
$\mathbf{c}$	25%	5	1	13.9%	4.6	0.60	920
$\mathbf{c}$	25%	5	5	22.9%	4.8	0.63	630
$\mathbf{b}_s$	100%	5	1	11.4%	4.0	0.85	560
$\mathbf{b}_s$	100%	5	5	11.9%	4.1	0.86	490

Table 9.1: Comparison between AAM algorithms given displaced centres (See Text)

Driven Params	Sub-sample	Iterations		Failure Rate	Final Errors	
		Max.	Forced		Point ± 0.1	Grey ± 0.01
$\mathbf{c}$	100%	5	1	3%	4.2	0.46
$\mathbf{c}$	100%	5	5	4%	4.4	0.47
$\mathbf{c}$	25%	5	1	10%	4.6	0.60
$\mathbf{c}$	25%	5	5	10%	4.6	0.60
$\mathbf{b}_s$	100%	5	1	6%	4.0	0.84
$\mathbf{b}_s$	100%	5	5	6%	4.1	0.87

Table 9.2: Comparison between AAM algorithms, given correct initial pose. (See Text)

9.4 Discussion

We have described several modifications that can be made to the Active Appearance Model algorithm. Sub-sampling and driving the shape parameters during search both lead to faster convergence, but were more prone to failure. The shape based method was able to locate the points slightly more accurately than the original formulation. Testing for improvement and convergence at each iteration slowed the search down, but lead to better final results.

It may be possible to use combinations of these approaches to achieve good results quickly, for instance using the shape based search in the early stages, then polishing with the original AAM. Further work will include developing strategies for reducing the numbers of convergence failures and extending the models to use colour or multispectral images.

Though only demonstrated for face models, the algorithm has wide applicability, for instance in matching models of structures in MR images [12]. The AAM algorithms, being able to match 10000 pixel, 100 parameter models to new images in a few seconds or less, are powerful new tools for image interpretation.

Chapter 10

Comparision between ASMs and AAMs

Given an appearance model of an object, we can match it to an image using either the Active Shape Model algorithm, or the Active Appearance Model algorithm. The ASM will find point locations only. Given these, it is easy to find the texture model parameters which best represent the texture at these points, and then the best fitting appearance model parameters.

The ASM matches the model points to a new image using an iterative technique which is a variant on the Expectation Maximisation algorithm. A search is made around the current position of each point to find a point nearby which best matches a model of the texture expected at the landmark. The parameters of the shape model controlling the point positions are then updated to move the model points closer to the points found in the image.

The AAM manipulates a full model of appearance, which represents both shape variation and the texture of the region covered by the model. This can be used to generate full synthetic images of modelled objects. The AAM uses the difference between the current synthesised image and the target image to update its parameters.

There are three key differences between the two algorithms:

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 under the current position.

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

This chapter describes results of experiments testing the performance of both ASMs and AAMs on two data sets, one of faces, the other of structures in MR brain sections. Measurements are made of their convergence properties, their accuracy in locating landmark points, their capture range and the time required to locate a target structure.

10.1 Experiments

Two data sets were used for the comparison:

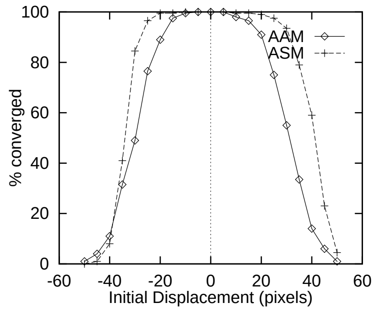
- 400 face images, each marked with 133 points
- 72 slices of MR images of brains, each marked up with 133 points around sub-cortical structures

For the faces, models were trained on 200 then tested on the remaining 200. For the brains leave-one-brain-out experiments were performed.

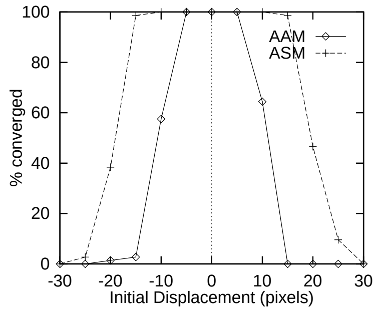
The Appearance model was built to represent 5000 pixels in both cases. Multi-resolution search was used, using 3 levels with resolutions of 25%, 50% and 100% of the original image in each dimension. At most 10 iterations were run at each resolution. The ASM used profile models 11 pixels long (5 either side of the point) at each resolution, and searched 3 pixels either side. The performance of the algorithms can depend on the choice of parameters - we have chosen values which have been found to work well on a variety of applications.

Capture range

The model instance was systematically displaced from the known best position by up to ± 100 pixels in x, then a search was performed to attempt to locate the target points. Figure 10.1 shows the RMS error in the position of the centre of gravity given the different starting positions for both ASMs and AAMs.



Face data



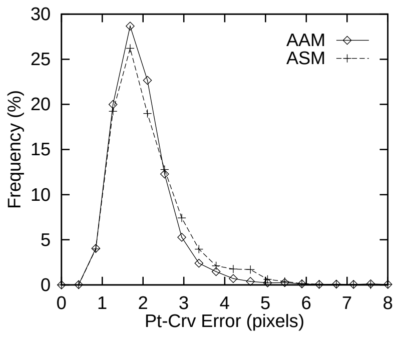
Brain data

Figure 10.1: Relative capture range of ASM and AAMs

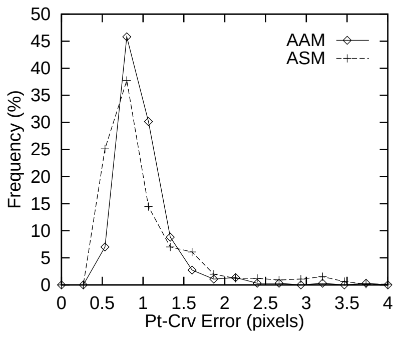
Thus the AAM has a slightly larger capture range for the face, but the ASM has a much larger capture range than the AAM for the brain structures. Of course, the results will depend on the resolutions used, the size of the models used and the search length of the ASM profiles.

Point location accuracy

One each test image we displaced the model instance from the true position by ± 10 in x and y (for the face) and ± 5 in x and y (for the brain), 9 displacements in total, then ran the search starting with the mean shape. On completion the results were compared with hand labelled points. Figure 10.2 shows frequency histograms for the resulting point-to-boundary errors(the distance from the found points to the associated boundary on the marked images). The ASM gives more accurate results than the AAM for the brain data, and comparable results for the face data.



Face data



Brain data

Figure 10.2: Histograms of point-boundary errors after search from displaced positions

Table 10.1 summarises the RMS point-to-point error, the RMS point-to-boundary error, the mean time per search and the proportion of convergence failures. Failure was declared when the RMS Point-Point error is greater than 10 pixels. The searches were performed on a 450MHz PentiumII PC running Linux.

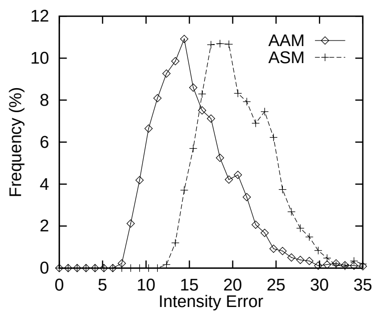
Data	Model	Time/search	Pt-Pt Error	Pt-Crv Error	Failures
Face	ASM	190ms	4.8	2.2	1.0%
	AAM	640ms	4.0	2.1	1.6%
Brain	ASM	220ms	2.2	1.1	0%
	AAM	320ms	2.3	1.1	0%

Table 10.1: Comparison of performance of ASM and AAM algorithms on face and brain data (See Text)

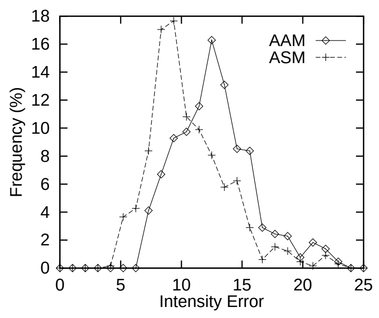
Thus the ASM runs significantly faster for both models, and locates the points more accurately than the AAM.

10.2 Texture Matching

The AAM explicitly generates a texture image which it seeks to match to the target image. After search we can thus measure the resulting RMS texture error. The ASM only locates points positions. However, given the points found by the ASM we can find the best fit of the texture model to the image, then record the residual. Figure 10.3 shows frequency histograms for the resulting RMS texture errors per pixel. The images have a contrast range of about [0,255]. The AAM produces a significantly better performance than the ASM on the face data, which is in part to be expected, since it is explicitly attempting to minimise the texture error. However, the ASM produces a better result on the brain data. This is caused by a combination of experimental set up and the additional constraints imposed by the appearance model.



Face data



Brain data

Figure 10.3: Histograms of RMS texture errors after search from displaced positions

Figure 10.4 compares the distribution of texture errors found after search with those obtained when the model is fit to the (hand marked) target points in the image (the 'Best Fit' line). This demonstrates that the AAM is able to achieve results much closer to the best fit results than the ASM (because it is more explicitly minimising texture errors). The difference between the best fit lines for the ASM and AAM has two causes;

- For the ASM experiments, though a leave-1-out approach was used for training the shape models and grey profile models, a single texture model trained on all the examples was used for the texture error evaluation. This could fit more accurately to the data than the model used by the AAM, trained in a leave-1-out regime.

- The AAM fits an appearance model which couples shape and texture explicitly - the ASM treats them as independent. For the relatively small training sets used this overconstrained the model, leading to poorer results.

The latter point is demonstrated in Figure 10.5, which shows the distribution of texture errors when fitting models to the training data. One line shows the errors when fitting a 50 mode texture model to the image (with shape defined by a 50 mode shape model fit to the labelled points). The second shows the best fit of a full 50 mode appearance model to the data. The additional constraints of the latter mean that for a given number of modes it is less able to fit to the data than independent shape and texture models, because the training set is not large enough to properly explore the variations. For a sufficiently large training set we would expect to be able to properly model the correlation between shape and texture, and thus be able to generate an appearance model which performed almost as well as a independent models, each with the same number of modes. Of course, if the total number of modes of the shape and texture model were constrained to that of the appearance model, the latter would perform much better.

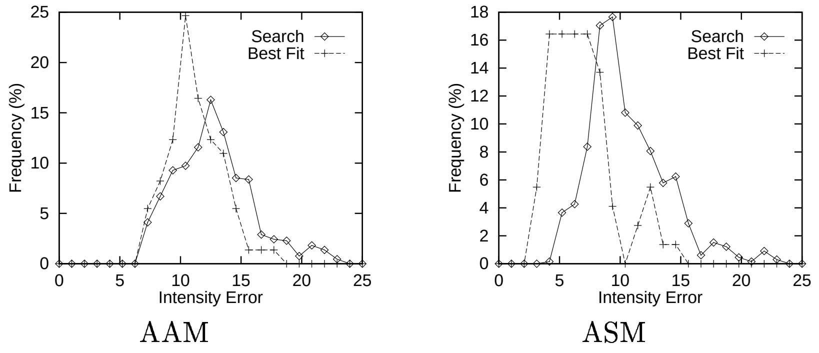


Figure 10.4: Histograms of RMS texture errors after search from displaced positions

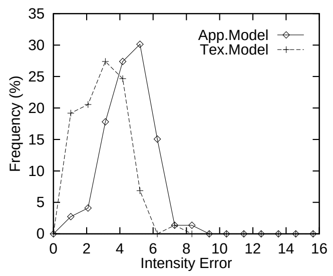


Figure 10.5: Comparison between texture model best fit and appearance model best fit

10.3 Discussion

Active Shape Models search around the current location, along profiles, so one would expect them to have a larger capture range than the AAM which only examines the image directly under its current area. This is clearly demonstrated in the results on the brain data set.

ASMs only use data around the model points, and do not take advantage of all the grey-level information available across an object as the AAM does. Thus they may be less reliable. However, the model points tend to be places of interest (boundaries or corners) where there is the most information. One could train an AAM to only search using information in areas near strong boundaries - this would require less image sampling during search so a potentially quicker algorithm. A more formal approach is to learn from the training set which pixels are most useful for search - this was explored in [13]. The resulting search is faster, but tends to be less reliable.

One advantage of the AAM is that one can build a convincing model with a relatively small number of landmarks. Any extra shape variation is expressed in additional modes of the texture model. The ASM needs points around boundaries so as to define suitable directions for search. Because of the considerable work required to get reliable image labelling, the fewer landmarks required, the better.

The AAM algorithm relies on finding a linear relationship between model parameter displacements and the induced texture error vector. However, we could augment the error vector with other measurements to give the algorithm more information. In particular one method of combining the ASM and AAM would be to search along profiles at each model point and augment the texture error vector with the distance along each profile of the best match. Like the texture error, this should be driven to zero for a good match. This approach will be the subject of further investigation.

To conclude, we find that the ASM is faster and achieves more accurate feature point location than the AAM. However, as it explicitly minimises texture errors the AAM gives a better match to the image texture.

Chapter 11

Further Developments

11.1 Automatic Landmark Placement

The most time consuming and scientifically unsatisfactory part of building shape models is the labelling of the training images. Manually placing hundreds (in 2D) or thousands (in 3D) of points on every image is both tedious and error prone. To reduce the burden, semi-automatic systems have been developed. In these a model is built from the current set of examples (possibly with extra artificial modes included in the early stages) and used to search the current image. The user can edit the result where necessary, then add the example to the training set. Though this can considerably reduce the time and effort required, labelling large sets of images is still difficult. It is particularly hard to place landmarks in 3D images, because of the difficulties of visualisation.

Ideally a fully automated system would be developed, in which the computer is presented with a set of training images and automatically places the landmarks to generate a model which is in some sense optimal. This is a difficult task, not least because it is not clear how to define what is optimal.

11.1.1 Automatic landmarking in 2D

Approaches to automatic landmark placement in 2D have assumed that contours (either pixellated or continuous, usually closed) have already been segmented from the training sets. The aim is to place points so as to build a model which best cptures the shape variation, but which has minimal representation error.

Baumberg and Hogg [2] describe a system which generates landmarks automatically for outlines of walking people. The outlines are represented as pixellated boundaries extracted automatically from a sequence of images using motion analysis. Landmarks are generated on an individual basis for each boundary by computing the principal axis of the boundary, identifying a reference pixel on the boundary at which the principal axis intersects the boundary and generating a number of equally spaced points from the reference point with respect to the path length of the boundary. While this process is satisfactory for silhouettes of pedestrians, it is unlikely that it will be generally successful. The authors went on to describe how the position of the landmarks can be iteratively updated in order to generate improved shape models generated from the landmarks [3].

Hill *et. al.*described a method of non-rigid correspondence in 2D between a pair of closed, pixellated boundaries [35] [36]. The method is based on generating sparse polygonal approximations for each shape; no curvature estimation for either boundary was required. The landmarks were further improved by an iterative refinement step. Results were presented which demonstrate the ability of this algorithm to provide accurate, non-rigid correspondences. This pair-wise corresponder was used within a framework for automatic landmark generation which demonstrated that landmarks similar to those identified manually were produced by this approach.

Kotcheff and Taylor [44] used direct optimisation to place landmarks on sets of closed curves. They define a mathematical expression which measures the compactness and the specificity of a model. This gives a measure which is a function of the landmark positions on the training set of curves. A genetic algorithm is used to adjust the point positions so as to optimise this measure.

Bookstein [6] describes an algorithm for landmarking sets of continuous contours represented as polygons. Points are allowed to slide along contours so as to minimise a bending energy term.

Rangarajan *et al* [59, 58] describe a method of point matching which simultaneously determines a set of matches and the similarity transform parameters required to register two contours defined by dense point sets. The method is robust against point features on one contour that do not appear on the other. An optimisation method similar to simulated annealing is used to solve the problem to produce a matrix of correspondences. The construction of the correspondence matrix cannot guarantee the production of a legal set of correspondences.

11.1.2 Automatic landmarking in 3D

The work on landmarking in 3D has mainly assumed a training set of closed 3D surfaces has already been segmented from the training images. As in 2D, the aim is to place landmarks across the set so as to give an 'optimal' model.

Fleute and Lavalée [25] use an framework of initially matching each training example to a single template, building a mean from these matched examples, and then iteratively matching each example to the current mean and repeating until convergence. Matching is performed using the multi-resolution registration method of Szeliski and Lavalée [69]. This method deforms the volume of space embedding the surface rather than deforming the surface itself. Kelemen *et al* [42] parameterise the surfaces of each of their shape examples using the method of Brechbühler *et al* [7]. Correspondence may then be established between surfaces but relies upon the choice of a parametric origin on each surface mapping and registration of the coordinate systems of these mappings by the computation of a rotation.

Brett *et. al.*[8] find correspondences on pairs of triangular meshes. A binary tree of corresponded shapes can be generated from a training set. Landmarks placed on the 'mean' shape at the root of the tree can be propogated to the leaves (the original training set). These landmarks are then used to build 3D shape models.

Caunce and Taylor [10] describe building 3D statistical models of the cortical sulci. Points are automatically located on the sulcal fissures and corresponded using variants on the Iterative Closest Point algorithm. The landmarks are progressively improved by adding in more structural and configural information. The final resulting landmarks are consistent with other anatomical studies.

Chapter 12

View-Based Appearance Models

The appearance of a face in a 2D image can change dramatically as the viewing angle changes. The majority of work on face tracking and recognition assumes near fronto-parallel views, and tends to break down when presented with large rotations or profile views. Three general approaches have been used to deal with this; a) use a full 3D model [71], b) introduce non-linearities into a 2D model [37] and c) use a set of models to represent appearance from different view points [52]. In this chapter we explore the last approach, using statistical models of shape and appearance to represent the variations in appearance from a particular viewpoint.

These appearance models are trained on example images labelled with sets of landmarks to define the correspondences between images [12]. Lanitis *et. al.*[47] showed that a linear model was sufficient to simulate considerable changes in viewpoint, as long as all the modelled features (the landmarks) remained visible. A model trained on near fronto-parallel face images can cope with pose variations of up to 45^o either side. For much larger angle displacements, some features become occluded, and the assumptions of the model break down.

We demonstrate that to deal with full 180^o rotation (from left profile to right profile), we need only 5 models, roughly centred on viewpoints at -90^o,-45^o,0^o,45^o,90^o (where 0^o corresponds to fronto-parallel). The pairs of models at $\pm 90^o$ (full profile) and $\pm 45^o$(half profile) are simply reflections of each other, so there are only 3 distinct models. We can use these models for estimating head pose, for tracking faces through wide changes in orientation and for synthesizing new views of a subject given a single view.

Each model is trained on labelled images of a variety of people with a range of orientations chosen so none of the features for that model become occluded. The different models use different sets of features (see Figure 12.1). Each example view can then be approximated using the appropriate appearance model with a vector of parameters, $\mathbf{c}$. We assume that as the orientation changes, the parameters, $\mathbf{c}$, trace out an approximately elliptical path. We can learn the relationship between $\mathbf{c}$ and head orientation, allowing us to both estimate the orientation of any head and to be able to synthesize a face at any orientation.

By using the Active Appearance Model algorithm [22, 12] we can match any of the individual models to a new image rapidly. If we know in advance the approximate pose, we can easily select the most suitable model. If we do not know, we can search with each of the five models and choose the one which achieves the best match. Once a model is selected and matched, we can estimate the head pose, and thus track the face, switching to a new model if the head pose varies significantly.

Given a single image of a new person, we can match the models to estimate the pose. We can then use the best fitting model to generate new views from angles similar to that of the original image. We can also exploit correlations across models of different views to estimate the appearance of the subject in a completely different view. Though this can perhaps be done most effectively with a full 3D model [71], we demonstrate that good results can be achieved just with a set of 2D models.

In the following we describe the techniques in more detail and give examples of the model, its ability to estimate pose, to track faces and to synthesize unseen views.

12.1 Related Work

Statistical models of shape and texture have been widely used for recognition, tracking and synthesis [39, 47, 22, 70], but have tended to only be used with near fronto-parallel images.

Moghaddam and Pentland [52] describe using view-based eigenface models to represent a wide variety of viewpoints. Our work is similar to this, but by including shape variation (rather than the rigid eigen-patches), we require fewer models and can obtain better reconstructions with fewer model modes.

Maurer and von der Malsburg [49] demonstrated tracking heads through wide angles by tracking graphs whose nodes are facial features, located with Gabor jets. The system is effective for tracking, but is not able to synthesize the appearance of the face being tracked.

Murase and Nayar [37] showed that the projections of multiple views of a rigid object into an eigenspace fell on a 2D manifold in that space. By modelling this manifold they could recognise objects from arbitrary views. A similar approach has been taken by Gong *et. al.*[62, 45] who use non-linear representations of the projections into an eigen-face space for tracking and pose estimation, and by Graham and Allinson [28] who use it for recognition from unfamiliar viewpoints.

Romdhani *et. al.*[60] have extended the Active Shape Model to deal with full 180^o rotation of a face using a non-linear model. However, the non-linearities mean the method is slow to match to a new image. They have also extended the AAM [67] using a kernel PCA. A non-linear 2D shape model is combined with a non-linear texture model on a 3D texture template. The approach is promising, but considerably more complex than using a small set of linear 2D models.

Vetter [71] has demonstrated how a 3D statistical model of face shape and texture can be used to generate new views given a single view. The model can be matched to a new image from more or less any viewpoint using a general optimisation scheme, though this is slow. Similar work has been described by Fua and Miccio [26] and Pighin *et. al.*[56]. By explicitly taking into account the 3D nature of the problem, this approach is likely to yield better reconstructions than the purely 2D method described below. However, the view based models we propose could be used to drive the parameters of the 3D head model, speeding up matching times.

12.2 Training Data

To explore the ability of the apperance models to represent the face from a range of angles, we gathered a training set consisting of sequences of individuals rotating their heads through 180^o, from full profile to full profile. Figure 12.1 shows typical examples, together with the landmark points placed on each example. The sets were divided up into 5 groups (left profile, left half profile, frontal, right half profile and right profile). Ambiguous examples were assigned to both groups, so that they could be used to learn the relationships between nearby views, allowing smooth transition between them (see below).

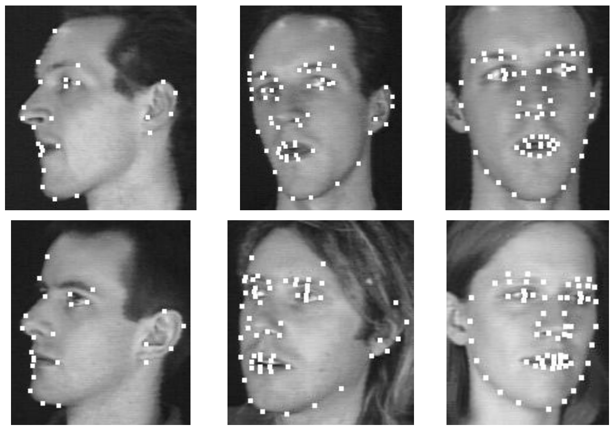


Figure 12.1: Examples from the training sets for the models

We trained three distinct models on data similar to that shown in Figure 12.1. The profile model was trained on 234 landmarked images taken of 15 individuals from different orientations. The half-profile model was trained on 82 images, and the frontal model on 294 images. Reflections of the images were used to enlarge the training set.

Figure 12.2 shows the effects of varying the first two appearance model parameters, c_1, c_2, of models trained on a set of face images, labelled as shown in Figure 12.1. These change both the shape and the texture component of the synthesised image.

12.3 Predicting Pose

We assume that the model parameters are related to the viewing angle, θ, approximately as

$$\mathbf{c} = \mathbf{c}_0 + \mathbf{c}_c \cos(\theta) + \mathbf{c}_s \sin(\theta) \tag{12.1}$$

where $\mathbf{c}_0$, $\mathbf{c}_c$ and $\mathbf{c}_s$ are vectors estimated from training data (see below).

(Here we consider only rotation about a vertical axis - head turning. Nodding can be dealt with in a similar way.)

This is an accurate representation of the relationship between the shape, $\mathbf{x}$, and orientation angle under an affine projection (the landmarks trace circles in 3D which are projected to ellipses

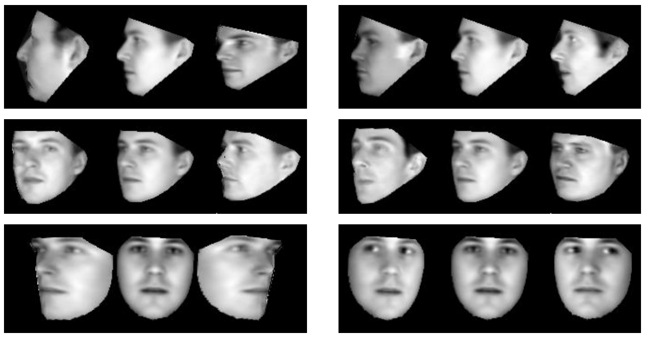


Figure 12.2: First two modes of the face models (top to bottom: profile, half-profile and frontal)

in 2D), but our experiments suggest it is also an acceptable approximation for the appearance model parameters, $\mathbf{c}$.

In order to learn the relationship for a given model, we must know the orientation of each of our training examples. We do not yet have access to a system which can measure it accurately, such as that used by [60, 45, 62]. However, we are able to estimate the angle by finding the frames in our training sequences at full profile and fronto-parallel by eye, then assuming a constant rate of rotation across the frames between. This leads to images labelled with orientations, θ_i, accurate to about $\pm 10^o$. For each such image we find the best fitting model parameters, $\mathbf{c}_i$. We then perform regression between $\{\mathbf{c}_i\}$ and the vectors $\{(1, \cos(\theta_i), \sin(\theta_i))'\}$ to learn $\mathbf{c}_0$,$\mathbf{c}_c$ and $\mathbf{c}_s$.

Figure 12.3 shows reconstructions in which the orientation, θ, is varied in Equation 12.1.

Given a new example with parameters $\mathbf{c}$, we can estimate its orientation as follows. Let $\mathbf{R}_c^{-1}$ be the left pseudo-inverse of the matrix $(\mathbf{c}_c|\mathbf{c}_s)$ (thus $\mathbf{R}_c^{-1}(\mathbf{c}_c|\mathbf{c}_s) = \mathbf{I}_2$).

Let

$$(x_a, y_a)' = \mathbf{R}_c^{-1}(\mathbf{c} - \mathbf{c}_0) \tag{12.2}$$

then the best estimate of the orientation is $\tan^{-1}(y_a/x_a)$.

Figure 12.4 shows the predicted orientations vs the actual orientations for the training sets for each of the models. It demonstrates that equation 12.1 is an acceptable model of parameter variation under rotation.

12.4 Tracking through wide angles

We can use the set of models to track faces through wide angle changes (full left profile to full right profile). We use a simple scheme in which we keep an estimate of the current head orientation and use it to choose which model should be used to match to the next image.

To track a face through a sequence we locate it in the first frame using a global search scheme similar to that described in [21]. This involves placing a model instance centred on each point on a grid across the image, then running a few iterations of the AAM algorithm. Poor fits are

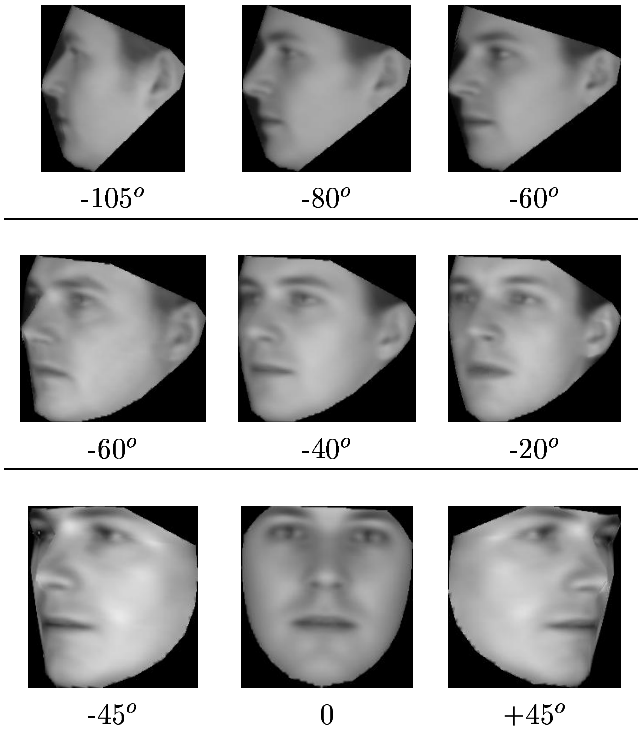


Figure 12.3: Rotation modes of three face models

discarded and good ones retained for more iterations. This is repeated for each model, and the best fitting model is used to estimate the position and orientation of the head.

We then project the current best model instance into the next frame and run a multi-resolution seach with the AAM. We estimate the head orientation from the results of the search, as described above. We then use the orientation to choose the most appropriate model with which to continue. Each model is valid over a particular range of angles, determined from its training set (see Table 12.1). If the orientation suggests changing to a new model, we estimate the parameters of the new model from those of the current best fit. We then perform an AAM search to match the new model more accurately. This process is repeated for each subsequent frame, switching to new models as the angle estimate dictates.

When switching to a new model we must estimate the image pose (position, within image orientation and scale) and model parameters of the new example from those of the old. We assume linear relationships which can be determined from the training sets for each model, as long as there are some images (with intermediate head orientations) which belong to the training sets for both models.

Figure 12.7 shows the results of using the models to track the face in a new test sequence (in this case a previously unseen sequence of a person who is in the training set). The model reconstruction is shown superimposed on frames from the sequence. The methods appears to track well, and is able to reconstruct a convincing simulation of the sequence.

We used this system to track 15 new sequences of the people in the training set. Each

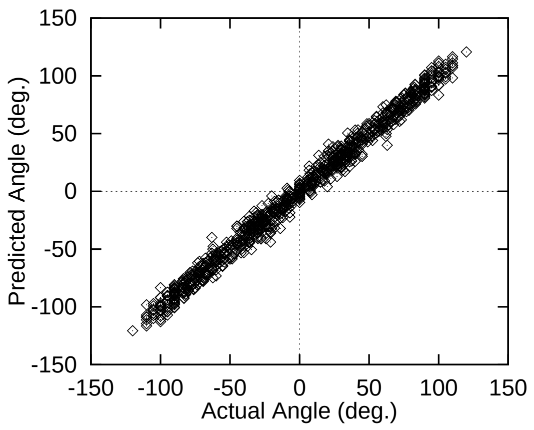


Figure 12.4: Prediction vs actual angle across training set

Model	Angle Range
Left Profile	-110^o - -60^o
Left Half-Profile	-60^o - -40^o
Frontal	-40^o - 40^o
Right Half-Profile	40^o - 60^o
Right Profile	60^o - 110^o

Table 12.1: Valid angle ranges for each model

sequence contained between 20 and 30 frames. Figure 12.5 shows the estimate of the angle from tracking against the actual angle. In all but one case the tracking succeeded, and a good estimate of the angle is obtained. In one case the models lost track and were unable to recover.

The system currently works off-line, loading sequences from disk. On a 450MHz Pentium III it runs at about 3 frames per second, though so far little work has been done to optimise this.

12.5 Predicting Unseen Views

Given a single view of a new person, we can find the best model match and determine their head orientation. We can then use the best model to synthesize new views at any orientation that can be represented by the model. If the best matching parameters are $\mathbf{c}$, we use equation 12.2 to estimate the angle, θ. Let $\mathbf{c}_{res}$ be the residual vector not explained by the rotation model, ie

$$\mathbf{c}_{res} = \mathbf{c} - (\mathbf{c}_0 + \mathbf{c}_c \cos(\theta) + \mathbf{c}_s \sin(\theta)) \tag{12.3}$$

To reconstruct at a new angle, α, we simply use the parameters

$$\mathbf{c}(\alpha) = \mathbf{c}_0 + \mathbf{c}_c \cos(\alpha) + \mathbf{c}_s \sin(\alpha) + \mathbf{c}_{res} \tag{12.4}$$

This only allows us to vary the angle in the range defined by the closest model. Since the models all represent the same 3D structure, we anticipate that there will be correlations between

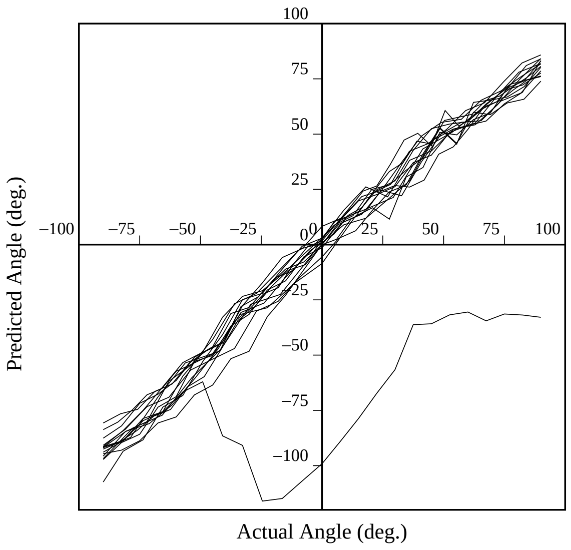


Figure 12.5: Comparison of angle derived from AAM tracking with actual angle (15 sequences)

parameters for different views of the same individual. To do this effectively we must first project out the effects of pose, lighting etc. A principled approach to this is described in [18]. However, for our experiments, since there is little lighting or expression change in the training set, it is sufficient just to remove the orientation components.

In order to learn the relationship between parameters in one model and those in another, we perform the following steps. For each frame in the training set we use equation 12.3 to determine the orientation independent component of the parameters for each model. We then compute the mean of such residuals for each person. Let $\bar{\mathbf{c}}_{i,j}$ be the mean of such residuals in the i^{th} model for the j^{th} person. By applying PCA to the means for a given model, we can find the projection, $\mathbf{P}_j$, into an 'identity' sub-space.

Let the projection of each mean in the subspace be

$$\mathbf{b}_{ij} = \mathbf{P}_j^T(\bar{\mathbf{c}}_{i,j} - \bar{\mathbf{c}}_j) \tag{12.5}$$

where $\bar{\mathbf{c}}_j$ is the mean of the means.

We can use linear regression to learn the relationship which maps each $\mathbf{b}_{ij}$ in the identity space of the j^{th} model to the corresponding mean $\mathbf{b}_{ik}$ in the identity space of the k^{th} model,

$$\mathbf{b}_{ij} = \mathbf{r}_{jk} + \mathbf{R}_{jk}\mathbf{b}_{ik} \tag{12.6}$$

Thus to reconstruct a new view of a person given a match in a different view;

1. remove the effects of orientation (Eq.12.3),
2. project into the identity sub-space for the model (Eq.12.5),
3. project across into the subspace of the target model (Eq.12.6),
4. project that into the residual space (inverting Eq.12.5)

5. add the appropriate orientation (Eq. 12.4).

Figure 12.6 demonstrates this. Models were built on the data for all but one person. The profile model was then matched to a profile image of the missing person (the reconstruction is shown). The method described above is then used to predict the appearance using the frontal model at two different angles. For comparison, corresponding images of the person at similar angles are shown. Given the small nature of the training set (in this case only 14 people, yielding a 13-D identity space), the results are encouraging.

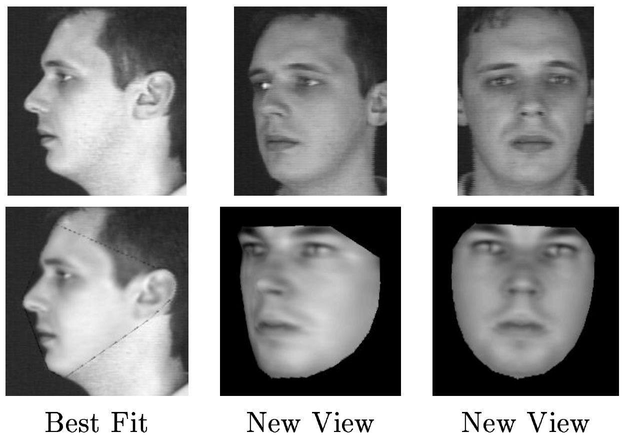


Figure 12.6: The best fit with a profile model is projected to the frontal model to predict new views

12.6 Discussion

This chapter demonstrates that a small number of view-based statistical models of appearance can represent the face from a wide range of viewing angles. Although we have concentrated on rotation about a vertical axis, rotation about a horizontal axis (nodding) could easily be included (and probably wouldn't require any extra models for modest rotations). We have shown that the models can be used to track faces through wide angle changes, and that they can be used to predict appearance from new viewpoints given a single image of a person.

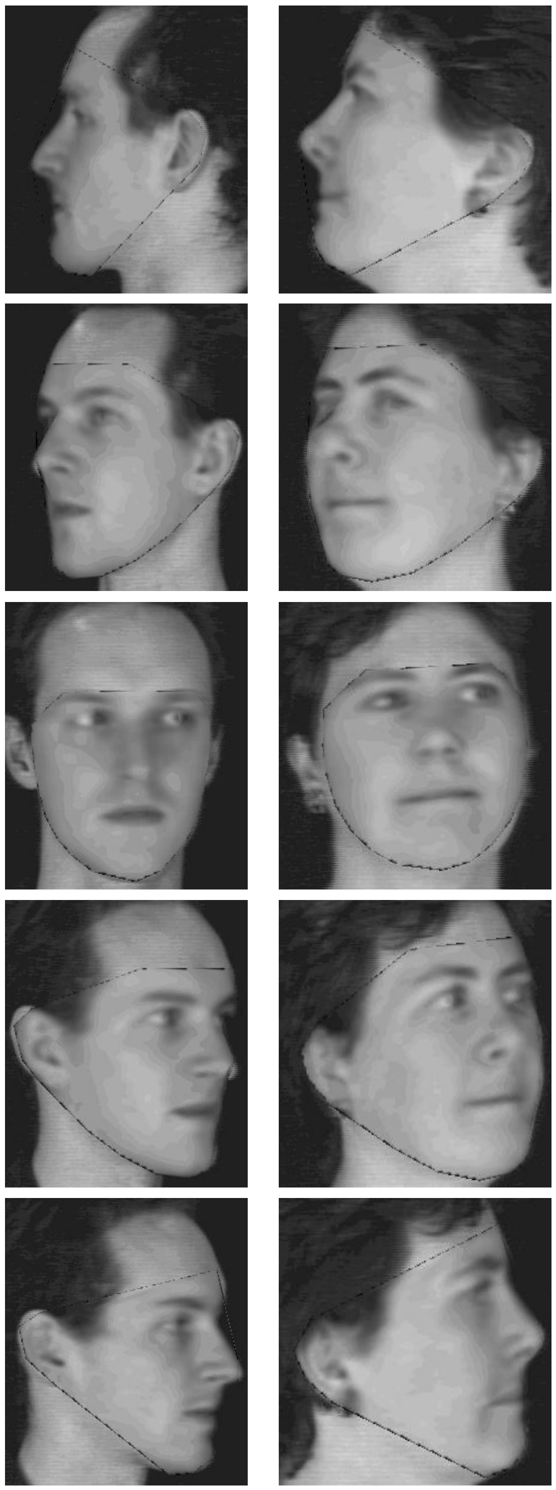


Figure 12.7: Reconstruction of tracked faces superimposed on sequences

Chapter 13

Discussion

Active Shape Models allow rapid location of the boundary of objects with similar shapes to those in a training set, assuming we know roughly where the object is in the image. They are particularly useful for:

- Objects with well defined shape (eg bones, organs, faces etc)
- Cases where we wish to classify objects by shape/appearance
- Cases where a representative set of examples is available
- Cases where we have a good guess as to where the target is in the image

However, they are not necessarily appropriate for

- Objects with widely varying shapes (eg amorphous things, trees, long wiggly worms etc)
- Problems involving counting large numbers of small things
- Problems in which position/size/orientation of targets is not known approximately (or cannot be easily estimated).

In addition, it should be noted that the accuracy to which they can locate a boundary is constrained by the model. The model can only deform in ways observed in the training set. If the object in an image exhibits a particular type of deformation not present in the training set, the model will not fit to it. This is true of fine deformations as well as coarse ones. For instance, the model will usually constrain boundaries to be smooth, if only smooth examples have been seen. Thus if a boundary in an image is rough, the model will remain smooth, and will not necessarily fit well. However, using enough training examples can usually overcome this problem.

One of the main drawbacks of the approach is the amount of labelled training examples required to build a good model. These can be very time consuming to generate. However, a 'bootstrap' approach can be adopted. We first annotate a single representative image with landmarks, and build a model from this. This will have a fixed shape, but will be allowed to scale, rotate and translate. We then use the ASM algorithm to match the model to a new image, and edit the points which do not fit well. We can then build a model from the two labelled examples we have, and use it to locate the points in the third. This process is repeated, incrementally building a model, until the ASM finds new examples sufficiently accurately every time, so needs no more training.

Both the shape models and the search algorithms can be extended to 3D. The landmark points become 3D points, the shape vectors become $3n$ dimensional for n points. Although the statistical machinery is identical, a 3D alignment algorithm must be used (see [33]). Of course, annotating 3D images with landmarks is difficult, and more points are required than for a 2D object. In addition, the definition of surfaces and 3D topology is more complex than that required for 2D boundaries. However, 3D models which represent shape deformation can be successfully used to locate structures in 3D datasets such as MR images (for instance [34]).

The ASM is well suited to tracking objects through image sequences. In the simplest form the full ASM search can be applied to the first image to locate the target. Assuming the object does not move by large amounts between frames, the shape for one frame can be used as the starting point for the search in the next, and only a few iterations will be required to lock on. More advanced techniques would involve applying a Kalman filter to predict the motion [2][23].

The shape models described above assume a simple gaussian model for the distribution of the shape parameters, $\mathbf{b}$. A more general approach is to use a mixture of gaussians. We need to ensure that the model generates plausible shapes. Using a single gaussian we can simply constrain the parameters with a bounding box or hyper-ellipse. With a mixture of gaussians we must arrange that the probability density for the current set of parameters is above a suitable threshold. Where it isn't, we can use gradient ascent to find the nearest point in parameter space which does give a plausible shape [16]. However, in practise, unless large non-gaussian shape variations are observed and the target images are noisy or cluttered, using a single gaussian approximation works perfectly well.

To summarise, by training statistical models of shape from sets of labelled examples we can represent both the mean shape of a class of objects and the common modes of shape variation. To locate similar objects in new images we can use the Active Shape Model algorithm which, given a reasonable starting point, can match the model to the image very quickly.

Acknowledgements

Dr Cootes is grateful to the EPSRC for his Advanced Fellowship Grant, and to his colleagues at the Wolfson Image Analysis Unit for their support. The MR cartilage images were provided by Dr C.E. Hutchinson and his team, and were annotated by Dr S.Solloway. The face images were annotated by G. Edwards, Dr A. Lanitis and other members of the Unit.

Appendix A

Applying a PCA when there are fewer samples than dimensions

Suppose we wish to apply a PCA to s n-D vectors, $\mathbf{x}_i$, where $s < n$. The covariance matrix is $n \times n$, which may be very large. However, we can calculate its eigenvectors and eigenvalues from a smaller $s \times s$ matrix derived from the data. Because the time taken for an eigenvector decomposition goes as the cube of the size of the matrix, this can give considerable savings.

Subract the mean from each data vector and put them into the matrix $\mathbf{D}$

$$\mathbf{D} = ((\mathbf{x}_1 - \bar{\mathbf{x}})|\dots|(\mathbf{x}_s - \bar{\mathbf{x}})) \tag{A.1}$$

The covariance matrix can be written

$$\mathbf{S} = \frac{1}{s}\mathbf{D}\mathbf{D}^T \tag{A.2}$$

Let $\mathbf{T}$ be the $s \times s$ matrix

$$\mathbf{T} = \frac{1}{s}\mathbf{D}^T\mathbf{D} \tag{A.3}$$

Let $\mathbf{e}_i$ be the s eigenvectors of $\mathbf{T}$ with corresponding eigenvalues λ_i, sorted into descending order. It can be shown that the s vectors $\mathbf{D}\mathbf{e}_i$ are all eigenvectors of $\mathbf{S}$ with corresponding eigenvalues λ_i, and that all remaining eigenvectors of $\mathbf{S}$ have zero eigenvalues. Note that $\mathbf{D}\mathbf{e}_i$ is not necessarily of unit length so may require normalising.

Appendix B

Aligning Two 2D Shapes

Given two 2D shapes, $\mathbf{x}$ and $\mathbf{x}'$, we wish to find the parameters of the transformation $T(.)$ which, when applied to $\mathbf{x}$, best aligns it with $\mathbf{x}'$. We will define 'best' as that which minimises the sum of squares distance. Thus we must choose the parameters so as to minimise

$$E = |T(\mathbf{x}) - \mathbf{x}'|^2 \tag{B.1}$$

Below we give the solution for the euclidean and the affine cases.

To simplify the notation, we define the following sums

$$\begin{array}{rclrcl} S_x & = & \frac{1}{n}\sum x_i & S_y & = & \frac{1}{n}\sum y_i \\ S_{x'} & = & \frac{1}{n}\sum x'_i & S_{y'} & = & \frac{1}{n}\sum y'_i \\ S_{xx} & = & \frac{1}{n}\sum x_i^2 & S_{yy} & = & \frac{1}{n}\sum y_i^2 \\ S_{xy} & = & \frac{1}{n}\sum x_i y_i & & & \\ S_{xx'} & = & \frac{1}{n}\sum x_i x'_i & S_{yy'} & = & \frac{1}{n}\sum y_i y'_i \\ S_{xy'} & = & \frac{1}{n}\sum x_i y'_i & S_{yx'} & = & \frac{1}{n}\sum y_i x'_i \end{array} \tag{B.2}$$

B.1 Euclidean Case

Suppose we have two shapes, $\mathbf{x}$ and $\mathbf{x}'$, centred on the origin ($\mathbf{x}.\mathbf{1} = \mathbf{x}'.\mathbf{1} = 0$). We wish to scale and rotate $\mathbf{x}$ by (s, θ) so as to minimise $|s\mathbf{A}\mathbf{x} - \mathbf{x}'|$, where $\mathbf{A}$ performs a rotation of a shape $\mathbf{x}$ by θ. Let

$$a = (\mathbf{x}.\mathbf{x}')/|\mathbf{x}|^2 \tag{B.3}$$

$$b = \left(\sum_{i=1}^{n}(x_i y'_i - y_i x'_i)\right)/|\mathbf{x}|^2 \tag{B.4}$$

Then $s^2 = a^2 + b^2$ and $\theta = \tan^{-1}(b/a)$. If the shapes do not have C.o.G.s on the origin, the optimal translation is chosen to match their C.o.G.s, the scaling and rotation chosen as above.

Proof

The two dimensional euclidean transformation is

$$T\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} a & -b \\ b & a \end{pmatrix}\begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} t_x \\ t_y \end{pmatrix} \tag{B.5}$$

We wish to find the parameters (a, b, t_x, t_y) of $T(.)$ which best aligns $\mathbf{x}$ with $\mathbf{x}'$, ie minimises

$$\begin{aligned} E(a,b,t_x,t_y) &= |T(\mathbf{x}) - \mathbf{x}')|^2 \\ &= \textstyle\sum_{i=1}^{n}(ax_i - by_i + t_x - x_i')^2 + (bx_i + ay_i + t_y - y_i')^2 \end{aligned} \tag{B.6}$$

Differentiating w.r.t. each parameter and equating to zero gives

$$\begin{aligned} a(S_{xx} + S_{yy}) + t_x S_x + t_y S_y &= S_{xx'} + S_{yy'} \\ b(S_{xx} + S_{yy}) + t_y S_x - t_x S_y &= S_{xy'} - S_{yx'} \\ aS_x - bS_y + t_x &= S_{x'} \\ bS_x + aS_y + t_y &= S_{y'} \end{aligned} \tag{B.7}$$

To simplify things (and without loss of generality), assume $\mathbf{x}$ is first translated so that its centre of gravity is on the origin. Thus $S_x = S_y = 0$, and we obtain

$$t_x = S_{x'} \qquad t_y = S_{y'} \tag{B.8}$$

and

$$\begin{aligned} a &= (S_{xx'} + S_{yy'})/(S_{xx} + S_{yy}) = \mathbf{x}.\mathbf{x}'/|\mathbf{x}|^2 \\ b &= (S_{xy'} - S_{yx'})/(S_{xx} + S_{yy}) = (S_{xy'} - S_{yx'})/|\mathbf{x}|^2 \end{aligned} \tag{B.9}$$

If the original $\mathbf{x}$ was not centred on the origin, then the initial translation to the origin must be taken into account in the final solution.

B.2 Affine Case

The two dimensional affine transformation is

$$T\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} a & b \\ c & d \end{pmatrix}\begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} t_x \\ t_y \end{pmatrix} \tag{B.10}$$

We wish to find the parameters (a, b, c, d, t_x, t_y) of $T(.)$ which best aligns $\mathbf{x}$ with $\mathbf{x}'$, ie minimises

$$\begin{aligned} E(a,b,c,d,t_x,t_y) &= |T(\mathbf{x}) - \mathbf{x}')|^2 \\ &= \textstyle\sum_{i=1}^{n}(ax_i + by_i + t_x - x_i')^2 + (cx_i + dy_i + t_y - y_i')^2 \end{aligned} \tag{B.11}$$

Differentiating w.r.t. each parameter and equating to zero gives

$$\begin{aligned} aS_{xx} + bS_{xy} + t_x S_x &= S_{xx'} & cS_{xx} + dS_{xy} + t_y S_x &= S_{xy'} \\ aS_{xy} + bS_{yy} + t_x S_y &= S_{yx'} & cS_{xy} + dS_{yy} + t_y S_y &= S_{yy'} \\ aS_x + bS_y + t_x &= S_{x'} & cS_x + dS_y + t_y &= S_{y'} \end{aligned} \tag{B.12}$$

where the S_* are as defined above.

To simplify things (and without loss of generality), assume $\mathbf{x}$ is first translated so that its centre of gravity is on the origin. Thus $S_x = S_y = 0$, and we obtain

$$t_x = S_{x'} \qquad t_y = S_{y'} \tag{B.13}$$

Substituting into B.12 and rearranging gives

$$\begin{pmatrix} a & b \\ c & d \end{pmatrix} \begin{pmatrix} S_{xx} & S_{xy} \\ S_{xy} & S_{yy} \end{pmatrix} = \begin{pmatrix} S_{xx'} & S_{yx'} \\ S_{xy'} & S_{yy'} \end{pmatrix} \tag{B.14}$$

Thus

$$\begin{pmatrix} a & b \\ c & d \end{pmatrix} = \frac{1}{\Delta} \begin{pmatrix} S_{xx'} & S_{yx'} \\ S_{xy'} & S_{yy'} \end{pmatrix} \begin{pmatrix} S_{yy} & -S_{xy} \\ -S_{xy} & S_{xx} \end{pmatrix} \tag{B.15}$$

where $\Delta = S_{xx}S_{yy} - S_{xy}^2$.

If the original $\mathbf{x}$ was not centred on the origin, then the initial translation to the origin must be taken into account in the final solution.

Appendix C

Aligning Shapes (II)

Here we present solutions to the problem of finding the optimal parameters to align two shapes so as to minimise a weighted sum-of-squares measure of point difference.

Throughout we assume two sets of n points, $\mathbf{x}_i$ and $\mathbf{x}'_i$. We assume a transformation, $\mathbf{x}' = T_{\mathbf{t}}(\mathbf{x})$ with parameters $\mathbf{t}$. We seek to choose the parameters, $\mathbf{t}$, so as to minimise

$$E = \sum_{i=1}^{n} (\mathbf{x}'_i - T_{\mathbf{t}}(\mathbf{x}_i))^T \mathbf{W}_i (\mathbf{x}'_i - T_{\mathbf{t}}(\mathbf{x}_i)) \tag{C.1}$$

The solutions are obtained by equating $\frac{\partial E}{\partial \mathbf{t}} = \mathbf{0}$ (detailed derivation is left to the interested reader).

C.1 Translation (2D)

If pure translation is allowed,

$$T_{\mathbf{t}}(\mathbf{x}) = \mathbf{x} + \begin{pmatrix} t_x \\ t_y \end{pmatrix} \tag{C.2}$$

and $\mathbf{t} = (t_x, t_y)^T$.

In this case the parameters (t_x, t_y) are given by the solution to the linear equation

$$\left(\sum_{i=1}^{n} W_i\right) \begin{pmatrix} t_x \\ t_y \end{pmatrix} = \left(\sum_{i=1}^{n} W_i (\mathbf{x}'_i - \mathbf{x}_i)\right) \tag{C.3}$$

For the special case with isotropic weights, in which $\mathbf{W}_i = w_i \mathbf{I}$, the solution is given by

$$\mathbf{t} = \left(\sum_{i=1}^{n} W_i\right)^{-1} \left(\sum_{i=1}^{n} w_i (\mathbf{x}'_i - \mathbf{x}_i)\right) \tag{C.4}$$

For the unweighted case the solution is simply the difference between the means,

$$\mathbf{t} = \bar{\mathbf{x}}' - \bar{\mathbf{x}} \tag{C.5}$$

C.2 Translation and Scaling (2D)

If translation and scaling are allowed,

$$T_{\mathbf{t}}(\mathbf{x}) = s\mathbf{x} + \begin{pmatrix} t_x \\ t_y \end{pmatrix} \tag{C.6}$$

and $\mathbf{t} = (s, t_x, t_y)^T$.

In this case the parameters (t_x, t_y) are given by the solution to the linear equation

$$\left(\begin{array}{c|c} S_{xWx} & \mathbf{S}_{Wx}^T \\ \hline \mathbf{S}_{Wx} & \mathbf{S}_W \\ & \end{array}\right) \left(\begin{array}{c} s \\ \hline t_x \\ t_y \end{array}\right) = \left(\begin{array}{c} S_{xWx'} \\ \hline \mathbf{S}_{Wx} \\ \\ \end{array}\right) \tag{C.7}$$

where

$$\begin{array}{lll} S_{xWx} = \sum \mathbf{x}_i^T \mathbf{W}_i \mathbf{x}_i & \mathbf{S}_{Wx} = \sum \mathbf{W}_i \mathbf{x}_i & \mathbf{S}_W = \sum \mathbf{W}_i \\ S_{xWx'} = \sum \mathbf{x}_i^T \mathbf{W}_i \mathbf{x}'_i & \mathbf{S}_{Wx'} = \sum \mathbf{W}_i \mathbf{x}'_i & \end{array} \tag{C.8}$$

In the unweighted case this simplifies to the solution of

$$\begin{pmatrix} S_{xx} + S_{yy} & S_x & S_y \\ S_x & n & 0 \\ S_y & 0 & n \end{pmatrix} \begin{pmatrix} s \\ t_x \\ t_y \end{pmatrix} = \begin{pmatrix} S_{xx'} + S_{yy'} \\ S_{x'} \\ S_{y'} \end{pmatrix} \tag{C.9}$$

where

$$\begin{array}{llll} S_{xx} = \sum x_i^2 & S_{yy} = \sum y_i^2 & S_x = \sum x_i & S_y = \sum y_i \\ S_{xx'} = \sum x_i x'_i & S_{yy'} = \sum y_i y'_i & S_{x'} = \sum x'_i & S_{y'} = \sum y'_i \end{array} \tag{C.10}$$

C.3 Euclidean (2D)

If translation, scaling and rotation are allowed,

$$T_{\mathbf{t}}(\mathbf{x}) = \begin{pmatrix} a & -b \\ b & a \end{pmatrix} \mathbf{x} + \begin{pmatrix} t_x \\ t_y \end{pmatrix} \tag{C.11}$$

and $\mathbf{t} = (a, b, t_x, t_y)^T$.

In this case the parameters (a, b, t_x, t_y) are given by the solution to the linear equation

$$\begin{pmatrix} S_{xWx} & S_{xWJx} & \mathbf{S}_{Wx}^T \\ S_{xWJx} & S_{xJWJx} & \mathbf{S}_{WJx}^T \\ \mathbf{S}_{Wx} & \mathbf{S}_{WJx} & \mathbf{S}_W \end{pmatrix} \begin{pmatrix} a \\ b \\ t_x \\ t_y \end{pmatrix} = \begin{pmatrix} S_{xWx'} \\ S_{xJWx'} \\ \mathbf{S}_{Wx'} \end{pmatrix} \tag{C.12}$$

where

$$\begin{array}{ll} S_{xJWJx} = \sum \mathbf{x}_i^T \mathbf{J}^T \mathbf{W}_i \mathbf{J} \mathbf{x}_i & \mathbf{S}_{WJx} = \sum \mathbf{W}_i \mathbf{J} \mathbf{x}_i \\ S_{xJWx'} = \sum \mathbf{x}_i^T \mathbf{J}^T \mathbf{W}_i \mathbf{x}'_i & \mathbf{S}_{WJx'} = \sum \mathbf{W}_i \mathbf{J} \mathbf{x}'_i \end{array} \tag{C.13}$$

and

$$\mathbf{J} = \begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix} \tag{C.14}$$

In the unweighted case this simplifies to the solution of

$$\begin{pmatrix} S_{xx}+S_{yy} & 0 & S_x & S_y \\ 0 & S_{xx}+S_{yy} & -S_y & S_x \\ S_x & -S_y & n & 0 \\ S_y & S_x & 0 & n \end{pmatrix} \begin{pmatrix} a \\ b \\ t_x \\ t_y \end{pmatrix} = \begin{pmatrix} S_{xx'}+S_{yy'} \\ S_{yx'}-S_{xy'} \\ S_{x'} \\ S_{y'} \end{pmatrix} \tag{C.15}$$

C.4 Affine (2D)

If a 2D affine transformation is allowed,

$$T_{\mathbf{t}}(\mathbf{x}) = \begin{pmatrix} a & b \\ c & d \end{pmatrix} \mathbf{x} + \begin{pmatrix} t_x \\ t_y \end{pmatrix} \tag{C.16}$$

and $\mathbf{t} = (a, b, c, d, t_x, t_y)^T$.

The anisotropic weights case is just big, but not complicated. I'll write it down one day.

In the unweighted case the parameters are given by the solution to the linear equation

$$\begin{pmatrix} S_{xx} & S_{xy} & S_x \\ S_{xy} & S_{yy} & S_y \\ S_x & S_y & n \end{pmatrix} \begin{pmatrix} a & c \\ b & d \\ t_x & t_y \end{pmatrix} = \begin{pmatrix} S_{xx'} & S_{xy'} \\ S_{yx'} & S_{yy'} \\ S_{x'} & S_{y'} \end{pmatrix} \tag{C.17}$$

Appendix D

Representing 2D Pose

D.1 Euclidean Case

For 2D shapes we usually allow translation (t_x, t_y), rotation, θ and scaling, s. To retain linearity we represent the pose using $\mathbf{t} = (s_x, s_y, t_x, t_y)^T$ where $s_x = (s\cos\theta - 1)$, $s_y = s\sin\theta$. In homogeneous co-ordinates, this corresponds to the transformation matrix

$$\mathbf{T_t} = \begin{pmatrix} 1+s_x & -s_y & t_x \\ s_y & 1+s_x & t_y \\ 0 & 0 & 1 \end{pmatrix} \tag{D.1}$$

For the AAM we must represent small changes in pose using a vector, $\delta\mathbf{t}$. This is to allow us to predict small pose changes using a linear regression model of the form $\delta\mathbf{t} = \mathbf{Rg}$. For linearity the zero vector should indicate no change, and the pose change should be approximately linear in the vector parameters. This is satisfied by the above parameterisation.

The AAM algorithm requires us to find the pose parameters $\mathbf{t}'$ of the transformation obtained by first applying the small change given by $\delta\mathbf{t}$, then the pose transform given by $\mathbf{t}$. Thus, find $\mathbf{t}'$ so that $T_{\mathbf{t}'}(\mathbf{x}) = T_{\mathbf{t}}(T_{\delta\mathbf{t}}(\mathbf{x}))$.

If we write $\delta\mathbf{t} = (\delta s_x, \delta s_y, \delta t_x, \delta t_y)^T$, it can be shown that

$$\begin{array}{rcl} 1+s'_x & = & (1+\delta s_x)(1+s_x) - \delta s_y s_y \\ s'_y & = & \delta t_2(1+s_x) + (1+\delta s_x)s_y \\ t'_x & = & (1+s_x)\delta t_x - s_y \delta t_y + t_x \\ t'_y & = & (1+s_x)\delta t_y + s_y \delta t_x + t_y \end{array} \tag{D.2}$$

Note that for small changes, $T_{\delta\mathbf{t}_1}(T_{\delta\mathbf{t}_2}(\mathbf{x})) \approx T_{(\delta\mathbf{t}_1 + \delta\mathbf{t}_2)}(\mathbf{x})$, which is required for the AAM predictions of pose change to be consistent.

Appendix E

Representing Texture Transformations

We allow scaling, α, and offsets, β, of the texture vectors $\mathbf{g}$. To retain linearity we represent the transformation parameters using the vector $\mathbf{u} = (u_1, u_2)^T = (\alpha - 1, \beta)^T$. Thus

$$T_{\mathbf{u}}(\mathbf{g}) = (1 + u_1)\mathbf{g} + u_2\mathbf{1} \tag{E.1}$$

As for the pose, the AAM algorithm requires us to find the parameters $\mathbf{u}'$ such that $T_{\mathbf{u}'}(\mathbf{g}) = T_{\mathbf{u}}(T_{\delta\mathbf{u}}(\mathbf{g}))$. It is simple to show that

$$\begin{aligned} 1 + u_1' &= (1 + u_1)(1 + \delta u_1) \\ u_2' &= (1 + u_1)\delta u_2 + u_2 \end{aligned} \tag{E.2}$$

Thus for small changes, $T_{\delta\mathbf{u}_1}(T_{\delta\mathbf{u}_2}(\mathbf{g})) \approx T_{(\delta\mathbf{u}_1 + \delta\mathbf{u}_2)}(\mathbf{g})$, which is required for the AAM predictions of pose change to be consistent.

Appendix F

Linear Regression

Given the t x s matrix $\mathbf{C}$ and the n x s matrix $\mathbf{X}$ we wish to find the t x n regression matrix $\mathbf{R}$ which satisfies

$$\mathbf{C} = \mathbf{R}\mathbf{X} \tag{F.1}$$

We assume that $n >> s > t$. The matrix $\mathbf{R}$ allows us to predict t-vectors, $\mathbf{c}$ from n-vectors, $\mathbf{x}$, using $\mathbf{c} = \mathbf{R}\mathbf{x}$.

Unfortunately the equation for $\mathbf{R}$ is hopelessly underdetermined and there are an infinite number of solutions.

F.1 The General Solution

Let the columns of $\mathbf{\Phi}_k$ be the k eigenvectors of $\mathbf{X}^T\mathbf{X}$ corresponding to the largest k eigenvalues $\{\lambda_1 ... \lambda_k\}$. Thus

$$\mathbf{X}^T\mathbf{X}\mathbf{\Phi}_k = \mathbf{\Phi}_k\mathbf{\Lambda}_k \tag{F.2}$$

where $\mathbf{\Lambda}_k = diag(\lambda_1 ... \lambda_k)$.

Let

$$\mathbf{B}_k = \mathbf{\Phi}_k^T = (\mathbf{\Lambda}_k^{-1}\mathbf{\Phi}_k^T\mathbf{X}^T)\mathbf{X} \tag{F.3}$$

The columns of $\mathbf{B}_k$ are thus a projection of the columns of $\mathbf{X}$ into a k dimensional subspace. A useful property is

$$\mathbf{B}_k\mathbf{B}_k^T = \mathbf{I}_k \tag{F.4}$$

Consider the equation

$$\mathbf{C} = \mathbf{R}'_k\mathbf{B}_k \tag{F.5}$$

If $\mathbf{R}'_k$ is a solution, then

$$\mathbf{R}_k = \mathbf{R}'_k(\mathbf{\Lambda}_k^{-1}\mathbf{\Phi}_k^T\mathbf{X}^T) \tag{F.6}$$

is a solution to (F.1).

Applying (F.4) to (F.5) we find

$$\mathbf{R}'_k = \mathbf{C}\mathbf{B}_k^T \tag{F.7}$$

and thus

$$\mathbf{R}_k = \mathbf{C}\mathbf{B}_k^T(\mathbf{\Lambda}_k^{-1}\mathbf{\Phi}_k^T\mathbf{X}^T) \tag{F.8}$$

The question remains, what is the optimal value for k, the number of modes to retain in the calculation? We can estimate this using a leave-one-out approach, and the power of linear algebra.

F.2 Optimal choice for k

Strictly we should find k by leaving out each column, j, of $\mathbf{C}$ and corresponding column of $\mathbf{X}$ in turn, estimating $\mathbf{R}_k$ and using it to predict the error in estimating $\mathbf{c}_j$ from $\mathbf{x}_j$. This is potentially very computationally intensive. A more tractable approach is to just miss out each column of $\mathbf{B}_k$ in turn. This can be done very efficiently.

Let $\mathbf{B}_{k/j}$ be a k x $(s-1)$ matrix equivalent to $\mathbf{B}_k$ with the j^{th} column, $\mathbf{b}_{kj}$, missing. Similarly let $\mathbf{C}_{/j}$ be $\mathbf{C}$ with the j^{th} column, $\mathbf{c}_j$ missing.

Now,

$$\begin{aligned} \mathbf{B}_{k/j}\mathbf{B}_{k/j}^T &= \mathbf{B}_k\mathbf{B}_k - \mathbf{b}_{kj}\mathbf{b}_{kj}^T \\ &= \mathbf{I}_k - \mathbf{b}_{kj}\mathbf{b}_{kj}^T \end{aligned} \tag{F.9}$$

It can be shown that

$$(\mathbf{I}_k - \mathbf{b}_{kj}^T\mathbf{b}_{kj})^{-1} = (\mathbf{I}_k + f_{kj}\mathbf{b}_{kj}^T\mathbf{b}_{kj}) \tag{F.10}$$

where

$$\begin{aligned} f_{kj} &= \tfrac{1}{1-\alpha_{kj}} \quad if\ |1-\alpha_{kj}| >= \epsilon \\ &= 0 \qquad\qquad\quad otherwise \end{aligned} \tag{F.11}$$

where $\alpha_{kj} = |\mathbf{b}_{kj}|^2$ and ϵ is a precision dependent value to avoid singularities.

Let $\mathbf{R}'_{k/j}$ be the solution to

$$\mathbf{C}_{/j} = \mathbf{R}'_{k/j}\mathbf{B}_{k/j} \tag{F.12}$$

Thus

$$\begin{aligned} \mathbf{R}'_{k/j}\mathbf{B}_{k/j} &= \mathbf{C}_{/j} \\ \mathbf{R}'_{k/j}\mathbf{B}_{k/j}\mathbf{B}_{k/j}^T &= \mathbf{C}_{/j}\mathbf{B}_{k/j}^T \\ \mathbf{R}'_{k/j} &= \mathbf{C}_{/j}\mathbf{B}_{k/j}^T(\mathbf{I}_k + f_{kj}\mathbf{b}_{kj}\mathbf{b}_{kj}^T) \end{aligned} \tag{F.13}$$

If we then use this to estimate $\mathbf{c}_j$ from $\mathbf{b}_{kj}$, the error is given by

$$\begin{aligned}\delta\mathbf{c}_{kj} &= \mathbf{R}'_{k/j}\mathbf{b}_{kj} - \mathbf{c}_j \\ &= \mathbf{C}_{/j}\mathbf{B}^T_{k/j}(\mathbf{I}_k + f_{kj}\mathbf{b}_{kj}\mathbf{b}^T_{kj})\mathbf{b}_{kj} - \mathbf{c}_j \\ &= \mathbf{C}_{/j}\mathbf{B}^T_{k/j}(\mathbf{b}_{kj} + \alpha_{kj}f_{kj}\mathbf{b}_{kj}) - \mathbf{c}_j \\ &= f_{kj}\mathbf{C}_{/j}\mathbf{B}^T_{k/j}\mathbf{b}_{kj} - \mathbf{c}_j\end{aligned} \tag{F.14}$$

Notice that $\mathbf{B}^T_{k/j}\mathbf{b}_{kj}$ is equal to the vector $\mathbf{B}^T_k\mathbf{b}_{kj}$ with the j^{th} element, α_{kj}, missing. Thus

$$\begin{aligned}\delta\mathbf{c}_{kj} &= f_{kj}(\mathbf{C}\mathbf{B}^T_k\mathbf{b}_{kj} - \alpha_{kj}\mathbf{c}_j) - \mathbf{c}_j \\ &= (\mathbf{C}\mathbf{B}^T_k\mathbf{b}_{kj} - \mathbf{c}_j)/(1-\alpha_{kj}) \qquad if|1-\alpha_{kj}| > \epsilon \\ &= -\mathbf{c}_j \qquad otherwise\end{aligned} \tag{F.15}$$

Thus to estimate the total leave-one-out error for k modes, calculate

$$E_k = \sum_j |\delta\mathbf{c}_k j|^2 \tag{F.16}$$

The optimal number of modes is that which minimises E_k. To ensure the resulting $\mathbf{R}$ spans the space, one should choose at least t modes.

Note: One method of calculating E_k is to compute $\delta\mathbf{C} = \mathbf{C}\mathbf{B}^T_k\mathbf{B}_k - \mathbf{C}$, then sum the square length of each column, weighted by f^2_{kj} (using $|\mathbf{c}_j|^2$ if $|1-\alpha_{kj}| \leq \epsilon$).

F.3 Example

Figure F.1 shows the plot of such a measure against number of modes for the regression used in the training of a face AAM.

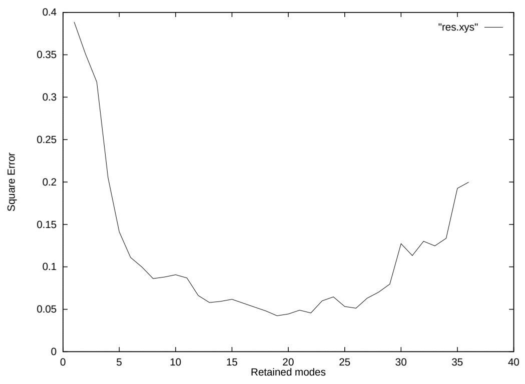


Figure F.1: Leave-one-out error vs number of modes retained (12 params, 42 displacements)

Appendix G

Image Warping

Suppose we wish to warp an image **I**, so that a set of n control points $\{ \mathbf{x}_i \}$ are mapped to new positions, $\{ \mathbf{x}'_i \}$. We require a continuous vector valued mapping function, $\mathbf{f}$, such that

$$\mathbf{f}(\mathbf{x}_i) = \mathbf{x}'_i \; \forall\, i = 1 \ldots n \tag{G.1}$$

Given such a function, we can project each pixel of image **I** into a new image $\mathbf{i}'$. In practice, in order to avoid holes and interpolation problems, it is better to find the reverse map, **f**', taking $\mathbf{x}'_i$ into $\mathbf{x}_i$. For each pixel in the target warped image, $\mathbf{i}'$ we can determine where it came from in **i** and fill it in. In general $\mathbf{f}' \neq \mathbf{f}^{-1}$, but is a good enough approximation.

Below we will consider two forms of $\mathbf{f}$, piece-wise affine and the thin plate spline interpolator.

Note that we can often break down $\mathbf{f}$ into a sum,

$$\mathbf{f}(\mathbf{x}) = \sum_{i=1}^{n} f_i(\mathbf{x})\mathbf{x}'_i \tag{G.2}$$

Where the n continuous scalar valued functions f_i each satisfy

$$f_i(\mathbf{x}_j) = \begin{matrix} 1 & if & i = j \\ 0 & & i \neq j \end{matrix} \tag{G.3}$$

This ensures $\mathbf{f}(\mathbf{x}_i) = \mathbf{x}'_i$.

G.1 Piece-wise Affine

The simplest warping function is to assume each f_i is linear in a local region and zero everywhere else.

For instance, in the one dimensional case (in which each $\mathbf{x}$ is a point on a line), suppose the control points are arranged in ascending order ($x_i < x_{i+1}$).

We would like to arrange that $\mathbf{f}$ will map a point $\mathbf{x}$ which is halfway between x_i and x_{i+1} to a point halfway between x'_i and x'_{i+1}. This is achieved by setting

$$f_i(x) = \begin{array}{ccl} (x-x_i)/(x_{i+1}-x_i) & if & x \in [x_i, x_{i+1}] \; and \; i < n \\ (x-x_i)/(x+i-x_{i-1}) & if & x \in [x_{i-1}, x_i] \; and \; i > 1 \\ 0 & & otherwise \end{array} \tag{G.4}$$

We can only sensibly warp in the region between the control points, $[x_1, x_n]$.

In two dimensions, we can use a triangulation (eg Delauney) to partition the convex hull of the control points into a set of triangles. To the points within each triangle we can apply the affine transformation which uniquely maps the corners of the triangle to their new positions in $\mathbf{i}'$.

Suppose $\mathbf{x}_1$, $\mathbf{x}_2$ and $\mathbf{x}_3$ are three corners of such a triangle. Any internal point can be written

$$\begin{aligned} \mathbf{x} &= \mathbf{x}_1 + \beta(\mathbf{x}_2 - \mathbf{x}_1) + \gamma(\mathbf{x}_3 - \mathbf{x}_1) \\ &= \alpha\mathbf{x}_1 + \beta\mathbf{x}_2 + \gamma\mathbf{x}_3 \end{aligned} \tag{G.5}$$

where $\alpha = 1 - (\beta + \gamma)$ and so $\alpha + \beta + \gamma = 1$. For $\mathbf{x}$ to be inside the triangle, $0 \leq \alpha, \beta, \gamma \leq 1$. Under the affine transformation, this point simply maps to

$$\mathbf{x}' = \mathbf{f}(\mathbf{x}) = \alpha\mathbf{x}'_1 + \beta\mathbf{x}'_2 + \gamma\mathbf{x}'_3 \tag{G.6}$$

To generate a warped image we take each pixel, $\mathbf{x}'$ in $\mathbf{I}'$, decide which triangle it belongs to, compute the coefficients α, β, γ giving its relative position in the triangle and use them to find the equivalent point in the original image, $\mathbf{I}$. We sample from this point and copy the value into pixel $\mathbf{x}'$ in $\mathbf{I}'$.

Note that although this gives a continuous deformation, it is not smooth. Straight lines can be kinked across boundaries between triangles (see Figure G.1).

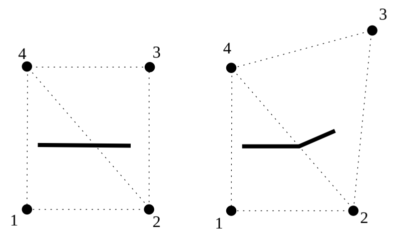


Figure G.1: Using piece-wise affine warping can lead to kinks in straight lines

G.2 Thin Plate Splines

Thin Plate Splines were popularised by Bookstein for statistical shape analysis and are widely used in computer graphics. They lead to smooth deformations, and have the added advantage over piee-wise affine that they are not constrained to the convex hull of the control points. However, they are more expensive to calculate.

G.3 One Dimension

First consider the one dimensional case. Let $U(r) = (\frac{r}{\sigma})^2 log(\frac{r}{\sigma})$, where σ is a scaling value defining the stiffness of the spline.

The 1D thin plate spline is then

$$f_1(x) = \sum_{i=1}^{n} w_i U(|x - x_i|) + a_0 + a_1 x \tag{G.7}$$

The weights w_i,a_0, a_1 are chosen to satisfy the constraints $f(x_i) = x'_i \, \forall i$.

If we define the vector function

$$\mathbf{u}_1(x) = (U(|x - x_1|), U(|x - x_2|, \ldots, U(|x - x_n|), 1, x)^T \tag{G.8}$$

and put the weights into a vector $\mathbf{w}_1 = (w_1, \ldots, w_n, a_0, a_1)$

then (G.7) becomes

$$f_1(x) = \mathbf{w}_1^T \mathbf{u}_1(x) \tag{G.9}$$

By plugging each pair of corresponding control points into (G.7) we get n linear equations of the form

$$x'_j = \sum_{i=1}^{n} w_i U(|x_j - x_i|) + a_0 + a_1 x_j \tag{G.10}$$

Let $U_i j = U_j i = U(|x_i - x_j|)$. Let $U_i i = 0$. Let $\mathbf{K}$ be the $n \times n$ matrix whose elements are $\{U_i j\}$.

Let

$$\mathbf{Q}_1 = \begin{pmatrix} 1 & x_1 \\ 1 & x_2 \\ \vdots & \vdots \\ 1 & x_n \end{pmatrix} \tag{G.11}$$

$$\mathbf{L}_1 = \begin{pmatrix} \mathbf{K} & \mathbf{Q}_1 \\ \mathbf{Q}_1^T & \mathbf{0}_2 \end{pmatrix} \tag{G.12}$$

where $(0)_2$ is a 2×2 zero matrix.

Let $\mathbf{X}'_1 = (x'_1, x'_2, \ldots, x'_n, 0, 0)^T$. Then the weights for the spline (G.7) $\mathbf{w}_1 = (w_1, \ldots, w_n, a_0, a_1)$ are given by the solution to the linear equation

$$\mathbf{L}_1 \mathbf{w} = \mathbf{X}'_1 \tag{G.13}$$

G.4 Many Dimensions

The extension to the d-dimensional case is straight forward.

If we define the vector function

$$\mathbf{u}_d(\mathbf{x}) = (U(|\mathbf{x} - \mathbf{x}_1|), \ldots, U(|\mathbf{x} - \mathbf{x}_n|), 1|\mathbf{x}^T)^T \tag{G.14}$$

then the d-dimensional thin plate spline is given by

$$\mathbf{f}(\mathbf{x}) = \mathbf{W}\mathbf{u}_d(\mathbf{x}) \tag{G.15}$$

where $\mathbf{W}$ is a $d \times (n + d + 1)$ matrix of weights.

To choose weights to satisfy the constraints, construct the matrices

$$\mathbf{Q}_d = \begin{pmatrix} 1 & \mathbf{x}_1^T \\ 1 & \mathbf{x}_2^T \\ \vdots & \vdots \\ 1 & \mathbf{x}_n^T \end{pmatrix} \tag{G.16}$$

$$\mathbf{L}_d = \begin{pmatrix} \mathbf{K} & \mathbf{Q}_d \\ \mathbf{Q}_d^T & \mathbf{0}_{d+1} \end{pmatrix} \tag{G.17}$$

where $\mathbf{0}_{d+1}$ is a $(d+1) \times (d+1)$ zero matrix, and $\mathbf{K}$ is a $n \times n$ matrix whose ij^{th} element is $U(|\mathbf{x}_i - \mathbf{x}_j|)$.

Then construct the $n + d + 1 \times d$ matrix $\mathbf{X}'_d$ from the positions of the control points in the warped image,

$$\mathbf{X}'_d = \begin{pmatrix} \mathbf{x}'_1 \\ \vdots \\ \mathbf{x}'_n \\ \mathbf{0}_d \\ \vdots \mathbf{0}_d \end{pmatrix} \tag{G.18}$$

The matrix of weights is given by the solution to the linear equation

$$\mathbf{L}_d^T \mathbf{W}_d^T = \mathbf{X}'_d \tag{G.19}$$

Note that care must be taken in the choice of σ to avoid ill conditioned equations.

Appendix H

Density Estimation

The kernel method of density estimation [63] gives an estimate of the p.d.f. from which N samples, $\mathbf{x}_i$, have been drawn as

$$p(\mathbf{x}) = \sum_{i=1}^{N} \frac{1}{Nh^d} K(\frac{\mathbf{x} - \mathbf{x}_i}{h}) \tag{H.1}$$

where $K(\mathbf{t})$ defines the shape of the kernel to be placed at each point, h is a smoothing parameter defining the width of each kernel and d is the dimension of the data. In general, the larger the number of samples, the smaller the width of the kernel at each point. We use a gaussian kernel with a covariance matrix equal to that of the original data set, $\mathbf{S}$, ie $K(\mathbf{t}) = G(\mathbf{t} : \mathbf{0}, \mathbf{S})$. The optimal smoothing parameter, h, can be determined by cross-validation [63].

H.1 The Adaptive Kernel Method

The adaptive kernel method generalises the kernel method by allowing the scale of the kernels to be different at different points. Essentially, broader kernels are used in areas of low density where few observations are expected. The simplest approach is as follows:

1. Construct a pilot estimate $p'(\mathbf{x})$ using (H.1).

2. Define *local bandwidth factors* $\lambda_i = (p'(\mathbf{x}_i)/g)^{-\frac{1}{2}}$, where g is the geometric mean of the $p'(\mathbf{x}_i)$

3. Define the *adaptive kernel estimate* to be

$$p(\mathbf{x}) = \frac{1}{N} \sum_{i=1}^{N} (h\lambda_i)^{-d} K(\frac{\mathbf{x} - \mathbf{x}_i}{h\lambda_i}) \tag{H.2}$$

H.2 Approximating the PDF from a Kernel Estimate

The kernel method can give a good estimate of the distribution. However, because it is constructed from a large number of kernels, it can be too expensive to use the estimate in an application. We wish to find a simpler approximation which will allow $p(\mathbf{x})$ to be calculated quickly.

We will use a weighted mixture of m gaussians to approximate the distribution derived from the kernel method.

$$p_{mix}(\mathbf{x}) = \sum_{j=1}^{m} w_j G(\mathbf{x} : \mu_j, \mathbf{S}_j) \tag{H.3}$$

Such a mixture can approximate any distribution up to arbitrary accuracy, assuming sufficient components are used. The hope is that a small number of components will give a 'good enough' estimate. The Expectation Maximisation (EM) algorithm [51] is the standard method of fitting such a mixture to a set of data. However, if we were to use as many components as samples ($m = N$), the optimal fit of the standard EM algorithm is to have a delta function at each sample point. This is unsatisfactory. We assume that the kernel estimate, $p_k(\mathbf{x})$ is in some sense an optimal estimate, designed to best generalise the given data. We would like $p_{mix}(\mathbf{x}) \rightarrow p_k(\mathbf{x})$ as $m \rightarrow N$.

A good approximation to this can be achieved by modifying the M-step in the EM algorithm to take into account the covariance about each data point suggested by the kernel estimate (see (H.3) below).

The number of gaussians used in the mixture should be chosen so as to achieve a given approximation error between $p_k(\mathbf{x})$ and $p_{mix}(\mathbf{x})$.

H.3 The Modified EM Algorithm

To fit a mixture of m gaussians to N samples $\mathbf{x}_i$, assuming a covariance of $\mathbf{T}_i$ at each sample, we iterate on the following 2 steps:

E-step Compute the contribution of the i^{th} sample to the j^{th} gaussian

$$p_{ij} = \frac{w_j G(\mathbf{x}_i : \mu_j, \mathbf{S}_j)}{\sum_{j=1}^{m} w_j G(\mathbf{x}_i : \mu_j, \mathbf{S}_j)} \tag{H.4}$$

M-step Compute the parameters of the gaussians,

$$w_j = \tfrac{1}{N} \textstyle\sum_i p_{ij} \quad , \quad \mu_j = \tfrac{1}{N w_j} \textstyle\sum_i p_{ij} \mathbf{x}_i \tag{H.5}$$

$$\mathbf{S}_j = \frac{1}{N w_j} \sum_i p_{ij} [(\mathbf{x}_i - \mu_j)(\mathbf{x}_i - \mu_j)^T + \mathbf{T}_i] \tag{H.6}$$

Strictly we ought to modify the E-step to take $\mathbf{T}_i$ into account as well, but in our experience just changing the M-step gives satisfactory results.

Appendix I

Implementation

Though the core mathematics of the models described above are relatively simple, a great deal of machinery is required to actually implement a flexible system. This could easily be done by a competent programmer.

However, implementations of the software are already available.

The simplest way to experiment is to obtain the MatLab package implementing the Active Shape Models, available from Visual Automation Ltd. This provides an application which allows users to annotate training images, to build models and to use those models to search new images. In addition the package allows limited programming via a MatLab interface.

A C++ software library, co-written by the author, is also available from Visual Automation Ltd. This allows new applications incorporating the ASMs to be written.

See `http://www.wiau.man.ac.uk/val.htm` for details of both of the above.

It is intended that a free (C++) software package implementing ASMs will be provided for the Image Understanding Environment (a free computer vision library of software). For more details and the latest status, see
`http://www.wiau.man.ac.uk/services/IUE/IUE_gate.html`.

In practice the algorithms work well on mid-range PC (200 Mhz). Search will usually take less than a second for models containing up to a few hundred points.

Details of other implementations will be posted on
`http://www.isbe.man.ac.uk`

Bibliography

[1] Bajcsy and A. Kovacic. Multiresolution elastic matching. *Computer Graphics and Image Processing*, 46:1–21, 1989.

[2] A. Baumberg and D. Hogg. Learning flexible models from image sequences. In J.-O. Eklundh, editor, 3^{nd} *European Conference on Computer Vision*, volume 1, pages 299–308. Springer-Verlag, Berlin, 1994.

[3] A. Baumberg and D. Hogg. An adaptive eigenshape model. In D. Pycock, editor, 6^{th} *British Machine Vison Conference*, pages 87–96. BMVA Press, Sept. 1995.

[4] M. J. Black and Y. Yacoob. Recognizing facial expressions under rigid and non-rigid facial motions. In 1^{st} *International Workshop on Automatic Face and Gesture Recognition 1995*, pages 12–17, Zurich, 1995.

[5] F. L. Bookstein. Principal warps: Thin-plate splines and the decomposition of deformations. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 11(6):567–585, 1989.

[6] F. L. Bookstein. Landmark methods for forms without landmarks: morphometrics of group differences in outline shape. *Medical Image Analysis*, 1(3):225–244, 1997.

[7] C. Brechbühler, G. Gerig, and O. Kübler. Parameterisation of closed surfaces for 3-D shape description. *Computer Vision, Graphics and Image Processing*, 61:154–170, 1995.

[8] A. D. Brett and C. J. Taylor. A framework for automated landmark generation for automated 3D statistical model construction. In 16^{th} *Conference on Information Processing in Medical Imaging*, pages 376–381, Visegrád, Hungary, June 1999.

[9] P. Burt. The pyramid as a structure for efficient computation. In A.Rosenfeld, editor, *Multi-Resolution Image Processing and Analysis*, pages 6–37. Springer-Verlag, Berlin, 1984.

[10] A. Caunce and C. J. Taylor. Using local geometry to build 3d sulcal models. In 16^{th} *Conference on Information Processing in Medical Imaging*, pages 196–209, 1999.

[11] G. E. Christensen, R. D. Rabbitt, M. I. Miller, S. C. Joshi, U. Grenander, T. A. Coogan, and D. C. V. Essen. Topological properties of smooth anatomic maps. In 14^{th} *Conference on Information Processing in Medical Imaging, France*, pages 101–112. Kluwer Academic Publishers, 1995.

[12] T. F. Cootes, G. J. Edwards, and C. J. Taylor. Active appearance models. In H.Burkhardt and B. Neumann, editors, 5^{th} *European Conference on Computer Vision*, volume 2, pages 484–498. Springer, Berlin, 1998.

[13] T. F. Cootes, G. J. Edwards, and C. J. Taylor. A comparative evaluation of active appearance model algorithms. In P. Lewis and M. Nixon, editors, 9^{th} *British Machine Vison Conference*, volume 2, pages 680–689, Southampton, UK, Sept. 1998. BMVA Press.

[14] T. F. Cootes and C. J. Taylor. Modelling object appearance using the grey-level surface. In E. Hancock, editor, 5th *British Machine Vison Conference*, pages 479–488, York, England, Sept. 1994. BMVA Press.

[15] T. F. Cootes and C. J. Taylor. Data driven refinement of active shape model search. In 7th *British Machine Vison Conference*, pages 383–392, Edinburgh, UK, 1996.

[16] T. F. Cootes and C. J. Taylor. A mixture model for representing shape variation. In A. Clarke, editor, 8th *British Machine Vison Conference*, pages 110–119. BMVA Press, Essex, Sept. 1997.

[17] T. F. Cootes, C. J. Taylor, D. Cooper, and J. Graham. Active shape models - their training and application. *Computer Vision and Image Understanding*, 61(1):38–59, Jan. 1995.

[18] N. Costen, T. F. Cootes, G. J. Edwards, and C. J. Taylor. Automatic extraction of the face identity subspace. In T. Pridmore and D. Elliman, editors, 10th *British Machine Vison Conference*, volume 1, pages 513–522, Nottingham, UK, Sept. 1999. BMVA Press.

[19] M. Covell. Eigen-points: Control-point location using principal component analysis. In 2nd *International Conference on Automatic Face and Gesture Recognition 1997*, pages 122–127, Killington, USA, 1996.

[20] I. Dryden and K. V. Mardia. *The Statistical Analysis of Shape.* Wiley, London, 1998.

[21] G. Edwards, T. F. Cootes, and C. J. Taylor. Advances in active appearance models. In 7th *International Conference on Computer Vision*, pages 137–142, 1999.

[22] G. Edwards, C. J. Taylor, and T. F. Cootes. Interpreting face images using active appearance models. In 3rd *International Conference on Automatic Face and Gesture Recognition 1998*, pages 300–305, Japan, 1998.

[23] G. J. Edwards, C. J. Taylor, and T. F. Cootes. Learning to identify and track faces in image sequences. In 8th *British Machine Vison Conference*, pages 130–139, Colchester, UK, 1997.

[24] T. Ezzat and T. Poggio. Facial analysis and synthesis using image-based models. In 2nd *International Conference on Automatic Face and Gesture Recognition 1997*, pages 116–121, Killington, Vermont, 1996.

[25] M. Fleute and S. Lavallee. Building a complete surface model from sparse data using statistical shape models: Application to computer assisted knee surgery. In *MICCAI*, pages 878–887, 1998.

[26] P. Fua and C. Miccio. From regular images to animated heads: A least squares approach. In H.Burkhardt and B. Neumann, editors, 5th *European Conference on Computer Vision*, volume 1, pages 188–202. Springer, Berlin, 1998.

[27] C. Goodall. Procrustes methods in the statistical analysis of shape. *Journal of the Royal Statistical Society B*, 53(2):285–339, 1991.

[28] D. Graham and N. Allinson. Face recognition from unfamiliar views: Subspace methods and pose dependency. In 3rd *International Conference on Automatic Face and Gesture Recognition 1998*, pages 348–353, Japan, 1998.

[29] U. Grenander and M. Miller. Representations of knowledge in complex systems. *Journal of the Royal Statistical Society B*, 56:249–603, 1993.

[30] J. Haslam, C. J. Taylor, and T. F. Cootes. A probabalistic fitness measure for deformable template models. In E. Hancock, editor, 5th *British Machine Vison Conference*, pages 33–42, York, England, Sept. 1994. BMVA Press, Sheffield.

[31] T. Heap and D. Hogg. Automated pivot location for the cartesian-polar hybrid point distribution model. In R. Fisher and E. Trucco, editors, 7th *British Machine Vison Conference*, pages 97–106, Edinburgh, UK, Sept. 1996. BMVA Press.

[32] A. Hill, T. F. Cootes, and C. J. Taylor. A generic system for image interpretation using flexible templates. In D. Hogg and R. Boyle, editors, 3rd *British Machine Vision Conference*, pages 276–285. Springer-Verlag, London, Sept. 1992.

[33] A. Hill, T. F. Cootes, and C. J. Taylor. Active shape models and the shape approximation problem. *Image and Vision Computing*, 14(8):601–607, Aug. 1996.

[34] A. Hill, T. F. Cootes, C. J. Taylor, and K. Lindley. Medical image interpretation: A generic approach using deformable templates. *Journal of Medical Informatics*, 19(1):47–59, 1994.

[35] A. Hill and C. J. Taylor. A method of non-rigid correspondence for automatic landmark identification. In 7th *British Machine Vison Conference*, pages 323–332. BMVA Press, Sept. 1996.

[36] A. Hill and C. J. Taylor. Automatic landmark identification using a new method of non-rigid correspondence. In 15th *Conference on Information Processing in Medical Imaging*, pages 483–488, 1997.

[37] H.Murase and S. Nayar. Learning and recognition of 3d objects from appearance. *International Journal of Computer Vision*, pages 5–25, Jan. 1995.

[38] M. J. Jones and T. Poggio. Multidimensional morphable models. In 6th *International Conference on Computer Vision*, pages 683–688, 1998.

[39] M. J. Jones and T. Poggio. Multidimensional morphable models : A framework for representing and matching object classes. *International Journal of Computer Vision*, 2(29):107–131, 1998.

[40] P. Karaolani, G. D. Sullivan, K. D. Baker, and M. J. Baines. A finite element method for deformable models. In 5th *Alvey Vison Conference, Reading, England*, pages 73–78, 1989.

[41] M. Kass, A. Witkin, and D. Terzopoulos. Snakes: Active contour models. In 1st *International Conference on Computer Vision*, pages 259–268, London, June 1987.

[42] A. Kelemen, G. Székely, and G. Guido Gerig. Three-dimensional Model-based Segmentation. Technical Report 178, Image Science Lab, ETH Zürich, 1997.

[43] M. Kirby and L. Sirovich. Appliction of the karhumen-loeve procedure for the characterization of human faces. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(1):103–108, 1990.

[44] A. C. W. Kotcheff and C. J. Taylor. Automatic construction of eigenshape models by direct optimisation. *Medical Image Analysis*, 2(4):303–314, 1998.

[45] J. Kwong and S. Gong. Learning support vector machines for a multi-view face model. In T. Pridmore and D. Elliman, editors, 10th *British Machine Vison Conference*, volume 2, pages 503–512, Nottingham, UK, Sept. 1999. BMVA Press.

[46] M. Lades, J. Vorbruggen, J. Buhmann, J. Lange, C. von der Malsburt, R. Wurtz, and W. Konen. Distortion invariant object recognition in the dynamic link architecture. *IEEE Transactions on Computers*, 42:300–311, 1993.

[47] A. Lanitis, C. J. Taylor, and T. F. Cootes. Automatic interpretation and coding of face images using flexible models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(7):743–756, 1997.

[48] K. V. Mardia, J. T. Kent, and A. N. Walder. Statistical shape models in image analysis. In E. Keramidas, editor, *Computer Science and Statistics: 23rd INTERFACE Symposium*, pages 550–557. Interface Foundation, Fairfax Station, 1991.

[49] T. Maurer and C. von der Malsburg. Tracking and learning graphs and pose on image sequences of faces. In 2nd *International Conference on Automatic Face and Gesture Recognition 1997*, pages 176–181, Los Alamitos, California, Oct. 1996. IEEE Computer Society Press.

[50] T. McInerney and D. Terzopoulos. Deformable models in medical image analysis: a survey. *Medical Image Analysis*, 1(2):91–108, 1996.

[51] G. McLachlan and K.E.Basford. *Mixture Models: Inference and Applications to Clustering.* Dekker, New York, 1988.

[52] B. Moghaddam and A. Pentland. Face recognition using view-based and modular eigenspaces. In *SPIE*, volume 2277, pages 12–21, 1994.

[53] C. Nastar and N. Ayache. Non-rigid motion analysis in medical images: a physically based approach. In 13th *Conference on Information Processing in Medical Imaging, Flagstaff, Arizona, USA*, pages 17–32. Springer-Verlag, 1993.

[54] C. Nastar, B. Moghaddam, and A. Pentland. Generalized image matching: Statistical learning of physically-based deformations. In 4th *European Conference on Computer Vision*, volume 1, pages 589–598, Cambridge, UK, 1996.

[55] A. P. Pentland and S. Sclaroff. Closed-form solutions for physically based modelling and recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(7):715–729, 1991.

[56] F. Pighin, R. Szeliski, and D. Salesin. Resynthesizing facial animation through 3d model-based tracking. In 7th *International Conference on Computer Vision*, pages 137–142, 1999.

[57] W. Press, S. Teukolsky, W. Vetterling, and B. Flannery. *Numerical Recipes in C (2nd Edition).* Cambridge University Press, 1992.

[58] A. Rangagajan, H. Chui, and F. L. Bookstein. The softassign procrustes matching algorithm. In 15th *Conference on Information Processing in Medical Imaging*, pages 29–42, 1997.

[59] A. Rangarajan, E. Mjolsness, S. Pappu, L. Davachi, P. S. Goldman-Rakic, and J. S. Duncan. A robust point matching algorithm for autoradiograph alignment. In *Visualisation in Biomedical Computing*, pages 277–286, 1996.

[60] S. Romdhani, S. Gong, and A. Psarrou. A multi-view non-linear active shape model using kernel pca. In T. Pridmore and D. Elliman, editors, 10th *British Machine Vison Conference*, volume 2, pages 483–492, Nottingham, UK, Sept. 1999. BMVA Press.

[61] S. Sclaroff and J. Isidoro. Active blobs. In 6th *International Conference on Computer Vision*, pages 1146–53, 1998.

[62] J. Sherrah, S. Gong, and E. Ong. Understanding pose discrimination in similarity space. In T. Pridmore and D. Elliman, editors, 10th *British Machine Vison Conference*, volume 2, pages 523–532, Nottingham, UK, Sept. 1999. BMVA Press.

[63] B. Silverman. *Density Estimation for Statistics and Data Analysis.* Chapman and Hall, London, 1986.

[64] S. Solloway, C. Hutchinson, J. Waterton, and C. J. Taylor. Quantification of articular cartilage from MR images using active shape models. In B. Buxton and R. Cipolla, editors, 4th *European Conference on Computer Vision*, volume 2, pages 400–411, Cambridge, England, April 1996. Springer-Verlag.

[65] P. Sozou, T. F. Cootes, C. J. Taylor, and E. DiMauro. Non-linear point distribution modelling using a multi-layer perceptron. In D. Pycock, editor, 6^{th} *British Machine Vison Conference*, pages 107–116, Birmingham, England, Sept. 1995. BMVA Press.

[66] P. Sozou, T. F. Cootes, C. J. Taylor, and E. D. Mauro. A non-linear generalisation of point distribution models using polynomial regression. *Image and Vision Computing*, 13(5):451–457, June 1995.

[67] S.Romdhani, A.Psarrou, and S. Gong. On utilising template and feature-based correspondence in multi-view appearance models. In 6^{th} *European Conference on Computer Vision*, volume 1, pages 799–813. Springer, 2000.

[68] L. H. Staib and J. S. Duncan. Boundary finding with parametrically deformable models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14(11):1061–1075, 1992.

[69] G. Szeliski and S. Lavalée. Matching 3-D anatomical surface with non-rigid deformations using octree-splines. *International Journal of Computer Vision*, 18(2):171–186, 1996.

[70] M. Turk and A. Pentland. Eigenfaces for recognition. *Journal of Cognitive Neuroscience*, 3(1):71–86, 1991.

[71] T. Vetter. Learning novel views to a single face image. In 2^{nd} *International Conference on Automatic Face and Gesture Recognition 1997*, pages 22–27, Los Alamitos, California, Oct. 1996. IEEE Computer Society Press.

[72] Y. Wang and L. H. Staib. Elastic model based non-rigid registration incorporating statistical shape information. In *MICCAI*, pages 1162–1173, 1998.

[73] A. L. Yuille, D. S. Cohen, and P. Hallinan. Feature extraction from faces using deformable templates. *International Journal of Computer Vision*, 8(2):99–112, 1992.