Automatic Methods for Training Statistical Models of Shape and Appearance

A thesis submitted to the University of Manchester

for the degree of Doctor of Philosophy

in the Faculty of Medicine, Dentistry and Nursing

2000

Kevin Walker

Division of Imaging Science and Biomedical Engineering

✓
Th21919

Contents

List of Figures

List of Tables

Abstract

In recent years much research has been devoted to the development of algorithms which model the shape and appearance of complex deformable objects. When combined with algorithms which can locate an instance of these models in a unseen images, image interpretation can be successfully performed. The models are typically created during a *training phase*, which involves gathering statistics from a set of training images containing examples of the object class. In order to capture statistics about the objects shape and appearance, a set of consistent landmarks are placed on each training example. This is typically done manually, a process which is time consuming and one which introduces human error into the statistics of the model.

This thesis explores automatic methods for training models of shape and appearance, task necessary if modeling techniques are to be employed by non-experts.

We tackle the problem using a feature based approach. In order to locate correspondences between a pair of training images, we attempt to locate features from the first image in the second. Some features have a greater probability of being correctly located than others. Features which have a high probability of being correctly matched are known as *salient features*. We develop two methods of evaluating the saliency of a feature resulting in a principled approach for automatically selecting which features to correspond.

Two approaches for training models automatically are developed. The first processes the training images in a *serial* manner, the second method makes no assump-

tions about the ordering of the training examples. The second method is known as a *parallel* approach. We show that both methods can build models automatically which are more compact than models trained using a manually generated correspondence. We also conclude that robust automatic model building approaches require a parallel approach.

Declaration

No portion of the work referred to in the thesis has been submitted in support of an application for another degree or qualification of this or any other university or other institute of learning.

Copyright

Acknowledgements

I would like to thank the following:

My supervisor Dr Tim Cootes for his guidance and encouragement throughout my research and thesis preparation. I also hold Tim responsible for my continued interest in Computer Vision.

Professor Chris Taylor, Dr Gareth Edwards, Dr Tim Parr, Dr Alan Brett and Dr Reyer Zwiggelaar for helpful technical discussions and support.

My fellow students, in particular Mike Rogers, Steve Caulkin, Anthony Holmes and Chris Wolstenholme, for making day to day life in the department enjoyable. And also for the endless games of football and nights out!

Warren Mittoo and the rest of the support staff for keeping all the computers up and running.

Jonny Ley for 13 years of friendship.

My mother Carole, and my sister Yvonne for their never-ending support and faith in everything I believe in.

About the Author

Kevin Walker received a first class BSc in computer science from Manchester University in 1997. In October 1997 he commenced the research in the Department of Medical Biophysics (now Imaging Science and Biomedical Engineering) also at the University of Manchester, for which this thesis is submitted. During this period he published the following papers related to the work in this thesis.

- K. N. Walker, T. Cootes, , and C. J. Taylor. Correspondence based on distinct points using image invariants. In 8th *British Machine Vision Conference*, pages 540–549, Colchester, UK, 1997.

- K. N. Walker, T. Cootes, , and C. J. Taylor. Locating salient object features. In P. Lewis and M. Nixon, editors, 9th *British Machine Vision Conference*, volume 2, pages 557–566, Southampton, UK, Sept. 1998. BMVA Press.

- K. N. Walker, T. F. Cootes, and C. J. Taylor. Locating salient facial features using image invariants. In 3rd *International Conference on Automatic Face and Gesture Recognition 1998*, pages 242–247, Nara, Japan, 1998.

- G. J. Edwards, T. F. Cootes, C. J. Taylor, and K. Walker. Advances in active appearance models. In *Proc. International Conference on Computer Vision*, pages 137–142, 1999.

- K. N. Walker, T. Cootes, , and C. J. Taylor. Automatically building appearance models from image sequences using salient features. In 10th *British Machine*

Vision Conference, volume 2, pages 463–472, 1999.

- K. N. Walker, T. F. Cootes and C. J. Taylor. Determining correspondences for statistical models of facial appearance. In 4^{rd} *International Conference on Automatic Face and Gesture Recognition 2000*, pages 272–276, 2000.

- T. F. Cootes, K. N. Walker and C. J. Taylor. View-based active appearance models. In 4^{th} *International Conference on Automatic Face and Gesture Recognition 2000*, pages 227–232, 2000.

- K. N. Walker, T. F. Cootes and C. J. Taylor. Determining Correspondences for statistical models of appearance. In *European Conference on Computer Vision*, 2000.

- K. N. Walker, T. Cootes and C. J. Taylor. Locating salient facial features using image invariants. *Image and Vision Computing, Face Recognition Special Issue.*, To Appear.

- K. N. Walker, T. Cootes and C. J. Taylor. Automatically building appearance models from image sequences using salient features. *Image and Vision Computing, BMVC 99 Special Issue.*, To Appear.

Chapter 1

Introduction

The work described in this thesis aims to locate a consistent set of landmarks across a set of training images where each image contains an example of a particular class of object. The motivation for this task lies in the popularity of model based schemes for interpreting images. Many of these schemes require a labeled set of training images in order to gather statistics about the shape and appearance of the object class. The labels define point correspondences between the training images. Typically the labels are provided by manually placing landmarks on consistent features in all training examples. Placing landmarks on all training images manually is both time consuming and error prone. More importantly this process requires knowledge of statistical modeling and hence prevents non-experts from taking advantage of this technology. This thesis explores algorithms which can automatically provide the correspondences across a training set of images necessary for building statistical models of appearance.

1.1 Statistical Models of Shape and Appearance

In recent years a great deal of research has been devoted to modeling the shape and appearance of specific object classes. Models of shape and appearance have proved to be powerful tools for interpreting images, particularly when combined with matching algorithms which allow the identification of model instances within new images [79, 25, 17, 56].

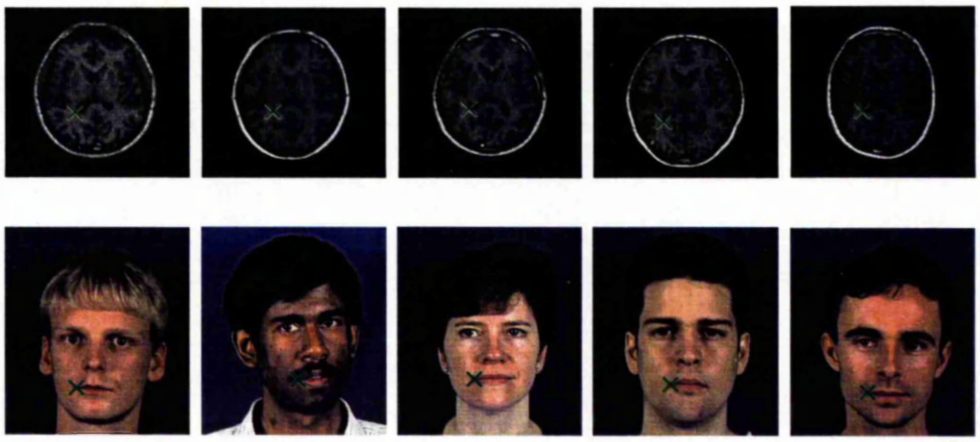


Figure 1.1: Each row illustrates how a landmark can be used to represent a point correspondence across a number of images of the same object class

Models are typically built by gathering statistics from a labeled training set of images of the object class. The process of gathering the statistics is called the *training phase*. Each training image has a number of landmarks where the i^{th} landmark in each image marks the position of the equivalent object feature. Figure 1.1 shows the position of one such landmark in several training images for different object classes. For rigid object classes with fixed 3D geometry 5 point correspondences are, in principle, sufficient to define the complete mapping between 2D projections. For more variable classes of object, such as faces, a much larger number of point correspondences may be required.

Once the model has been trained, search algorithms can be used repeatedly to locate further instances of the object class in a new image. Figures 1.2, 1.3 and 1.4 show examples of this for several object classes. Once the object is located, the information gathered about its shape and appearance can be used to interpret the

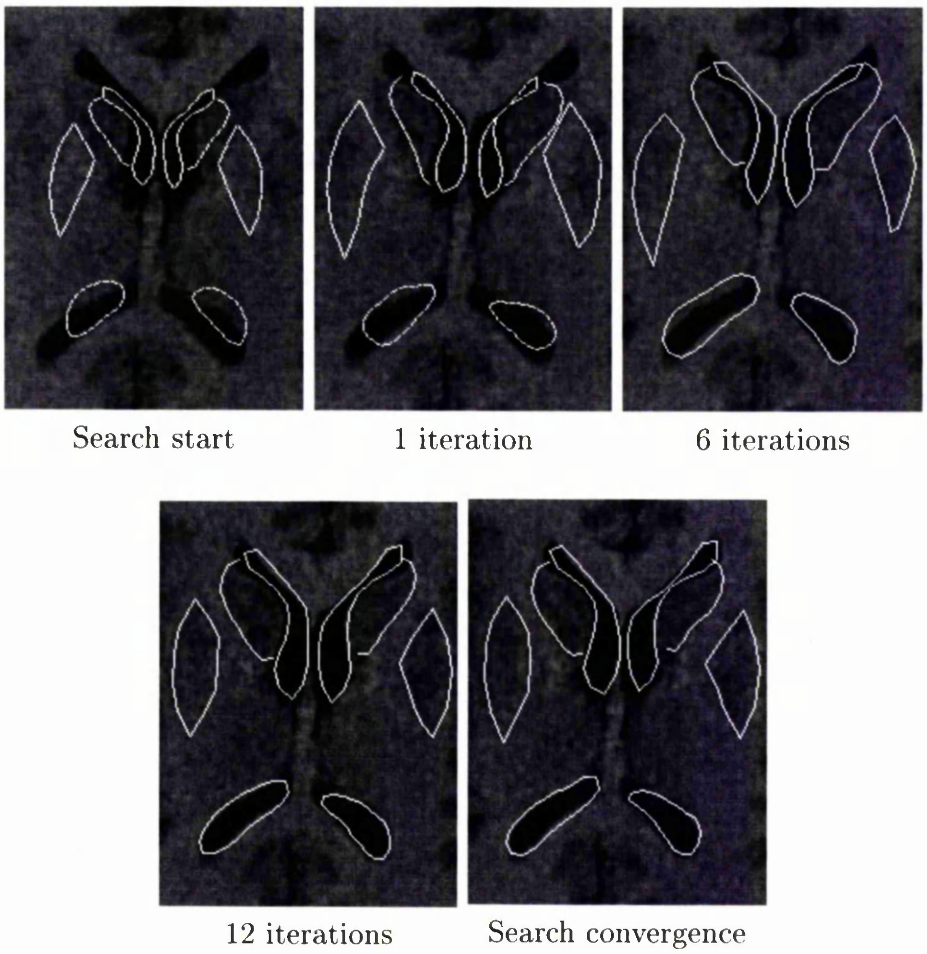


Figure 1.2: An example showing a statistical model of shape locating structures within the human brain.

object. For the example of a face, interpretation could involve any of the follow tasks:

- *face identification*, identifying exactly who the person is.
- *expression recognition*, what expression is the person displaying? Are they happy, sad, angry etc.
- *pose estimation*, what direction is the head facing?
- *gaze estimation*, what direction are the eyes looking?

All of these tasks can be solved if the shape and appearance of an unseen face is

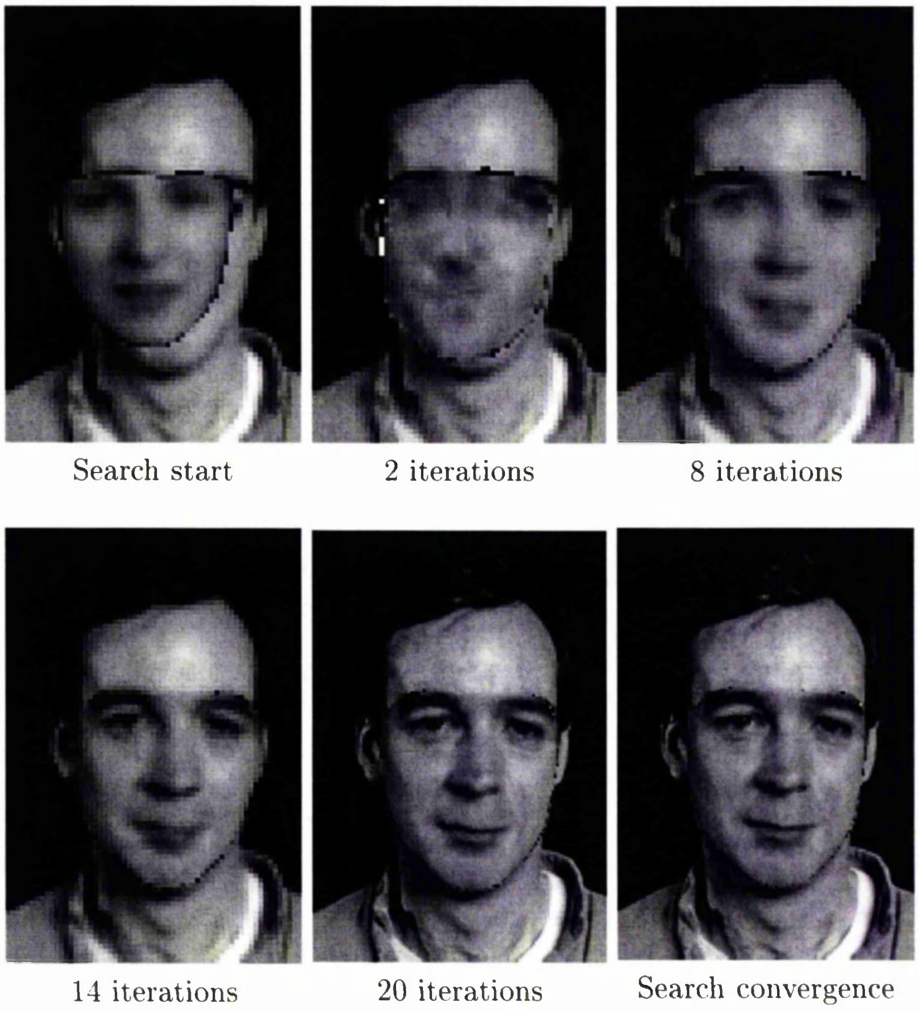


Figure 1.3: An example showing a statistical model of shape and appearance matching to an image of a human face.

accurately recovered.

The most time consuming and scientifically unsatisfactory part of building the models is the labeling of the training images as this is typically done manually. Manually placing hundreds of points on every image is both tedious and error prone.

Typically it requires the system designer to select an initial subset of features, a task which is subjective but critical to the success of the system. This task is normally performed simply by selecting features which correspond to corners, edges

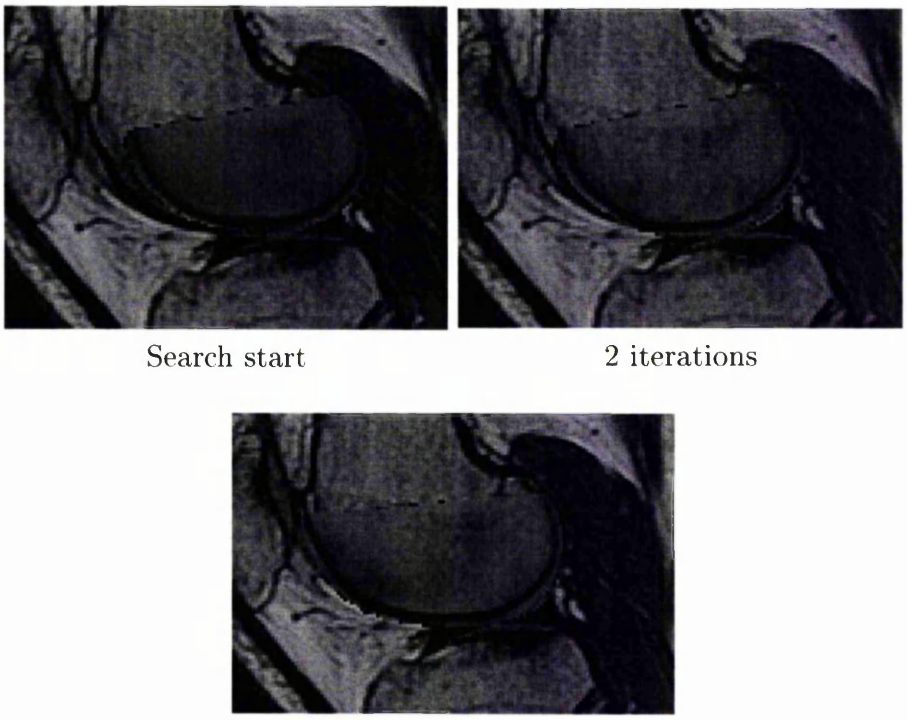


Figure 1.4: An example showing a statistical model of shape and appearance locating the position of the Femoral Articular Cartilage from a Magnetic Resonance image of a human knee.

or highly structured regions.

In this thesis we aim to develop principled methods which automatically calculate the correspondences necessary to build statistical models of an object's shape and its appearance. This would make the process of building models largely automatic, eliminating the possibility of human error and ensuring the resulting models are well suited for their purpose. Further more, it would make statistical modeling a valuable tool for the much wider audience of non-experts.

1.2 Approach

We tackle the problem using a feature based approach. The first problem we address is how to match a pair of images. We attempt to locate features from the first image in the second. Some features have a greater probability of being correctly located than others. Features which have a high probability of being correctly matched are known as *salient features.* We develop two methods of evaluating the saliency of a feature resulting in a principled approach for automatically selecting which features to correspond. The first method is known as *image feature saliency* and computes feature saliency based on a single image example. The second called *object feature saliency* computes feature saliency from a number of training images for which a correspondence already exists.

Two approaches for training models automatically are developed. The first processes the training images in a *serial* manner, the second method makes no assumptions about the ordering of the training examples. The second method is known as a *parallel* approach. We show that both methods can build models automatically which are more compact than models trained using a manually generated correspondence. We also conclude that robust automatic model building approaches require a parallel approach.

1.3 Outline of Thesis

Chapter 2 reviews statistical models of shape and appearance and their use in image interpretation. In particular we review point distribution models (PDM), active shape models (ASM) and active appearance models (AAM). It is the automatic construction of these types of models which is the main aim of this thesis.

Chapter 3 reviews current approaches to automatically computing correspondences over image sets. We describe several approaches, concentrating on those most closely related to the method we chose.

Chapter 4 reviews a number of local feature descriptors and how they can be used to measure the similarity between various image regions.

Chapter 5 describes image feature saliency and how it can be used to select the most distinctive (salient) features from within an image. We compare these features with features selected manually to see how accurately they can be located in a second similar image. We also review literature that has benefited from the notion of saliency.

Chapter 6 describes object feature saliency. Like image feature saliency, object feature saliency can be used to locate the positions of the most distinctive features within an object. Unlike image feature saliency, object feature saliency examines multiple image examples for which a correspondence already exists to assess the saliency of a feature.

Chapter 7 describes a method for building models automatically from sequences of images. The approach is based on a tracking framework which utilises image feature saliency.

Chapter 8 describes an approach to building models from image sets. We show how globally inconsistent transforms calculated between all image pairs can be used in an iterative scheme which calculates the required globally consistent transform across the entire image set.

Chapter 9 describes a multi resolution extension to the method described in Chapter 8. We demonstrate that the multi resolution framework increases the system's robustness to more extreme variation in the training images.

Chapter 10 contains a general discussion of the work presented in this thesis and possible directions of future research.

Chapter 2

Interpreting Images with Statistical Models

In this chapter we will look at a number of techniques in which shape and appearance can be modeled, and also how these models can be used by search schemes to locate new instances of the object in unseen images. We will concentrate on methods related to those used later in the thesis.

2.1 Modeling Shape

We are interested in interpreting objects, such as faces, whose shape can under go a wide range of deformations. A good model of object shape will encapsulate the object's range of legal deformations in a compact manner, whilst not allowing illegal deformations. Cootes *et al* addressed this problem by proposing *point distribution models* (PDMs). PDMs are statistical models of shape variation trained from a set of example images of the object. The following sections describe the construction of a PDM. For detailed descriptions of the statistics of shape, the reader is referred to Dryden and Mardia [31].

2.1.1 Labeling the training set

The PDM is a statistical model of the shape variations displayed in a training set. The training set consists of 2D images from a particular view of a 3D object. Before the statistics can be gathered, the shape displayed in each training image must be described in a consistent manner. This is achieved by placing a consistent list of n landmark points in each training example. A particular landmark will annotate the equivalent feature in each training image, for example, landmark 3 might represent the position of the center of the left eye in all training images. The shape described by training image i can be represented by vector $\boldsymbol{x}_i$:

$$\boldsymbol{x}_i = (x_1, x_2, ..., x_{n_l}, y_1, y_2, ..., y_{n_l})^T \tag{2.1}$$

where (x_j, y_j) is the position of the j^{th} landmark and n_l is the number of landmarks in each image.

The choice of which features should be annotated is an important one. We require features which are easily identified in all examples of the object.

Bookstein [9] defines three principal types of landmarks:

- *Type 1: discrete juxtapositions*, this type includes points in space which three or more structures meet. An example of this type of landmark is the branching points of tree like structures.
- *Type 2: maxima of curvature*, these include the tips of extrusions and valleys, such as the end of the nose on a human face.
- *Type 3: extremal points.* Bookstein defines extremal points as points the definitions of which refer to information at diverse separated locations.

This type includes evenly spaced radial intercepts, centroids, intersections of interlandmark segments and endpoints.

Figure 2.1 shows examples of face images annotated with landmarks.

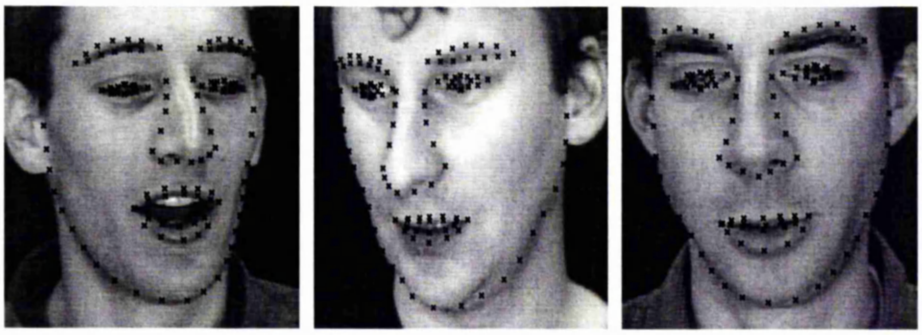


Figure 2.1: Consistent landmarks placed on images of faces.

2.1.2 Aligning the training set

We require a PDM which encodes all face specific variation but not variation due to the face's relative position in the image frame. Therefore variation due to scale, in-plane rotation and translation is not required in the model. For this reason the sets of points defining the shape are aligned, removing any such variation from the model. In order to do this the Generalised Procrustes Analysis method [40] is adopted. This method aligns each shape so that the sum of distances of each shape to the mean is minimised. Further details of the alignment procedure are given by Cootes *et al* [25].

2.1.3 Modeling the shape variation

The Procustes Analysis results in a set of aligned training shape vectors, $\boldsymbol{x}_i$ with a dimensionality of $2n_l$. The number of degrees of freedom in which the shapes can vary is typically much less than $2n_l$. This is because the ways in which the landmarks move between examples is highly correlated. For example, landmarks placed around the edge of the nose will always form roughly a 'U' shaped configuration. A PDM uses *Principal Component Analysis* [61] in order to capture these correlations and therefore reduce the number of shape parameters. The approach is as follows:

The mean shape, $\bar{\boldsymbol{x}}$, is given by:

$$\bar{\boldsymbol{x}} = \frac{1}{n_t} \sum_{i=1}^{n_t} \boldsymbol{x}_i \tag{2.2}$$

where n_t is the number of training examples.

The $2n_l \times 2n_l$ covariance matrix, $\boldsymbol{S}$, of the data is:

$$\boldsymbol{S} = \frac{1}{s-1} \sum_{i=1}^{s} (\boldsymbol{x}_i - \bar{\boldsymbol{x}})(\boldsymbol{x}_i - \bar{\boldsymbol{x}})^T \tag{2.3}$$

The training shape data, $\boldsymbol{x}_i$, form a cloud of points in a $2n_l$-D space. The eigenvectors, $\boldsymbol{p}_j$, and corresponding eigenvalues, λ_j $(j = 1, \ldots, (2n_l - 1))$, of $\boldsymbol{S}$ represent a set of orthogonal axis which are aligned with the principal modes of variation of the cloud. The eigenvectors corresponding to the largest eigenvalues represent the most significant modes along which the shape can vary. The eigenvalue λ_j gives the variance along the j^{th} component. Most of the shape variation can be represented by selecting a small number, n_e, of these axes which explain the largest amount of variation. Often n_e is chosen so the selected axes explain at least, say 95%, of the variance exhibited in the training set.

Any shape, $\boldsymbol{x}$, in the training set can then be approximated by a weighted sum of the first n_e eigenvectors and the mean shape:

$$\boldsymbol{x} \approx \bar{\boldsymbol{x}} + \boldsymbol{P}\boldsymbol{b}_s \tag{2.4}$$

where $\boldsymbol{P} = (\boldsymbol{p}_1, \boldsymbol{p}_2, \ldots \boldsymbol{p}_{n_e})$ is the matrix of the first n_e eigenvectors, and $\boldsymbol{b}_s$ is a n_e dimensional vector of weights, normally referred to as *shape parameters.*

The shape parameters, $\boldsymbol{b}_s$, which best match the model to a particular shape vector, $\boldsymbol{x}$, can be calculated as follows:

$$\boldsymbol{b}_s = \boldsymbol{P}^T(\boldsymbol{x} - \bar{\boldsymbol{x}}) \tag{2.5}$$

$\boldsymbol{b}_s$ defines a set of n_e model parameters. By varying a combination of these parameters we can generate new instances of the object's shape using equation 2.4. In order to prevent shapes which differ dramatically from those displayed in the training set, limits are applied to each of the shape parameters. The i^{th} parameter, b_i, is typically given limits $\pm 3\sqrt{\lambda_i}$. Figure 2.2 illustrates varying the 3 most significant modes for a PDM trained on examples of a human face.

2.2 Modeling Appearance

The shape variation is only part of the object's complete variation. The remaining variation can be encoded in a model of grey-level (or colour) appearance. Turk and Pentland [82] proposed a popular technique for modeling the grey levels of human faces known as *eigenfaces.* This technique aligned all the training images so they are normalised for translation, scale and in-plane rotation. The aligned images are represented as vectors and Principal Component Analysis (PCA) is performed, ensuring a low dimensional representation of the training data. This technique suffered because

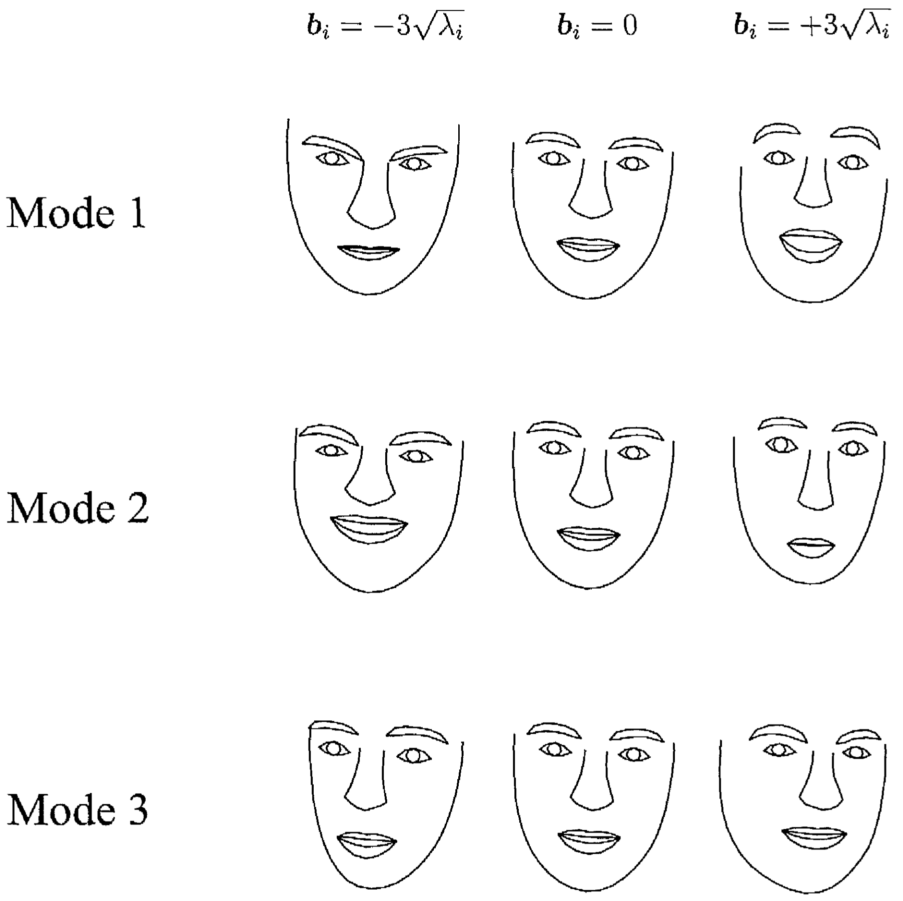


Figure 2.2: Effect of varying each of first three face shape parameters between ± 3 s.d.

the correspondence between training examples did not account for the natural shape variability in faces, such as introduced by expression, pose and identity variation. This problem was addressed by Craw *et al* [27] by transforming each training image into a *shape free space* before PCA is performed. The resulting model is known as a *shape-free region model.* In the following we explain how a shape-free region model is constructed and how it can be used to generate synthetic images.

2.2.1 Normalising Shape Variation

Given a labeled and aligned training set (Sections 2.1.1 and 2.1.2) we require further processing to normalise for the natural variability of theobject's shape. In this procedure, we deform each training image to a standard shape. After deformation the landmarks for each training image coincide. This is achieved by using a warping algorithm to 'morph' each training example to the mean shape as defined by a PDM (see equation 2.2). Generally piece-wise affine warping is used, although in certain circumstances a smoother alternative such as thin-plate splines as described by Bookstein is preferable [9]. Figure 2.3 shows the effect of warping three faces to the mean shape using a piece-wise affine warper (see Appendix A for further details of image warping). The resulting images are known as *shape-free* patches.

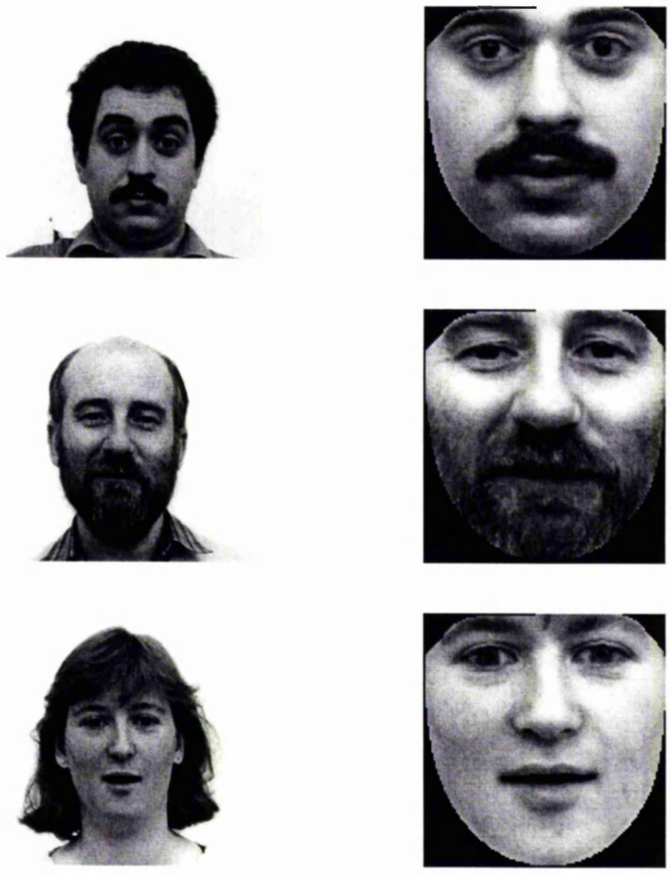


Figure 2.3: Example faces with extracted 'shape-free' patches.

2.2.2 Modeling Shape-Free Variation

Because the shape-free patches are all of a standard shape, each patch can be represented as an n_p dimensional vector of grey levels as follows:

$$\boldsymbol{g} = (g_1, g_2, ..., g_{n_p})^T \tag{2.6}$$

where g_i is the grey level intensity of the i^{th} pixel in the shape-free patch and n_p is the number of pixels represented . For colour images a $3n_p$ element vector would be needed. A model of the shape-free patches can be constructed by applying Principal Component Analysis to the grey-level vectors of the form:

$$\boldsymbol{g} \approx \bar{\boldsymbol{g}} + \boldsymbol{P}_g \boldsymbol{b}_g \tag{2.7}$$

where $\bar{\boldsymbol{g}}$ is the mean grey-level vector, $\boldsymbol{P}_g$ is a matrix of t_g eigenvectors corresponding to the largest eigenvalues and $\boldsymbol{b}_g$ is the is a vector of grey-level parameters. t_g is selected so that at least, say 95%, of the grey-level variance is described by the model. The eigenvectors selected in $\boldsymbol{P}_g$ describe a set of orthogonal *modes of grey-level variation*. We can calculate the grey-level parameters, $\boldsymbol{b}_g$, for a given grey-level vector, $\boldsymbol{g}$, as follows

$$\boldsymbol{b}_g = \boldsymbol{P}_g^T(\boldsymbol{g} - \bar{\boldsymbol{g}}) \tag{2.8}$$

By varying elements of $\boldsymbol{b}_g$ between a set of limits we can use equation 2.7 to reconstruct new shape-free patches. Figure 2.4 * show the effect of varying the three most significant grey-level parameters independently.

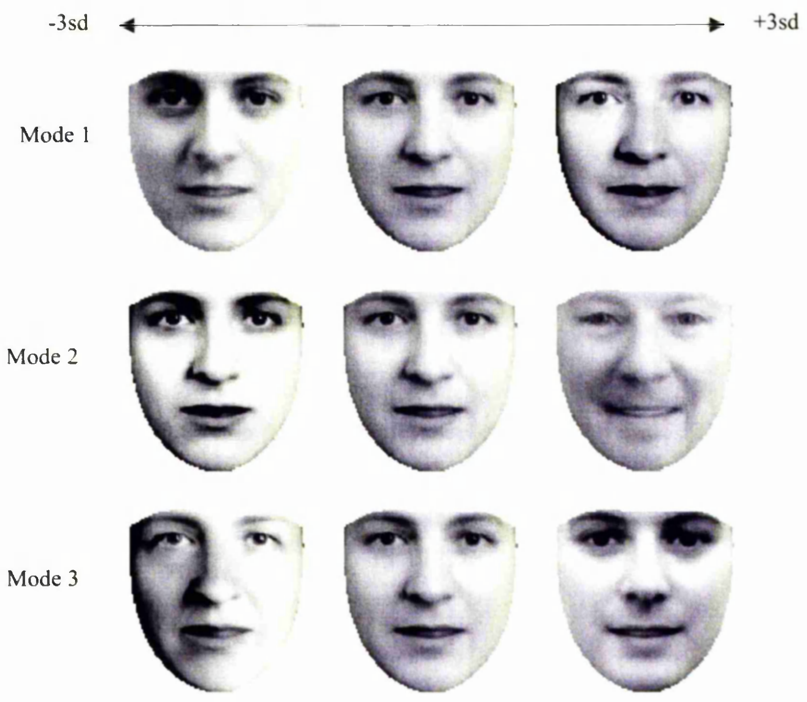


Figure 2.4: First three modes of variation of a typical shape-free face model.

2.3 Combined Models of Shape and Appearance

In sections 2.1 and 2.2 we described how the shape and appearance of any training example could be summarised by the parameter vectors $\boldsymbol{b}_s$ and $\boldsymbol{b}_g$. These described the shape and appearance in an independent manner. There may be, however, correlations between the shape and appearance of an object. If these correlations could be learnt the specificity of the model would be improved. Edwards *et al* [34] proposed

*This illustration was kindly provided by Dr. G Edwards a *Combined Appearance Model.* The Combined appearance model applied a further PCA to the shape parameters and grey-level parameters. For each training image we generate a concatenated vector:

$$\boldsymbol{b} = \begin{pmatrix} \boldsymbol{W}_s \boldsymbol{b}_s \\ \boldsymbol{b}_g \end{pmatrix} = \begin{pmatrix} \boldsymbol{W}_s \boldsymbol{P}_s^T (\boldsymbol{x} - \bar{\boldsymbol{x}}) \\ \boldsymbol{P}_g^T (\boldsymbol{g} - \bar{\boldsymbol{g}}) \end{pmatrix} \tag{2.9}$$

where $\boldsymbol{W}_s$ is a diagonal matrix of weights for each shape parameter. This is necessary because the shape and grey-level parameters are measured in different units. We apply a PCA on the set of all $\boldsymbol{b}$ parameters to give us the combined model:

$$\boldsymbol{b} \approx \boldsymbol{Q}\boldsymbol{c} \tag{2.10}$$

where $\boldsymbol{Q}$ is a matrix of t eigenvectors corresponding to the largest eigenvalues and $\boldsymbol{c}$ is the is a vector of combined appearance parameters which control both the shape and grey-level appearance of the model. t is selected so that a given proportion, say 98%, of the total variance is described by the model.

Since the columns of $\boldsymbol{Q}$ are orthogonal we can obtain $\boldsymbol{c}$ from $\boldsymbol{b}$ using

$$\boldsymbol{c} = \boldsymbol{Q}^T \boldsymbol{b} \tag{2.11}$$

Because of the linear relationship of the model we can express the shape and grey-levels directly as a functions of $\boldsymbol{c}$

$$\boldsymbol{x} \approx \bar{\boldsymbol{x}} + \boldsymbol{P}_s \boldsymbol{W}_s^{-1} \boldsymbol{Q}_s \boldsymbol{c} \tag{2.12}$$

$$\boldsymbol{g} \approx \bar{\boldsymbol{g}} + \boldsymbol{P}_g \boldsymbol{Q}_g \boldsymbol{c} \tag{2.13}$$

where

$$\boldsymbol{Q} = \begin{pmatrix} \boldsymbol{Q}_s \\ \boldsymbol{Q}_g \end{pmatrix} \tag{2.14}$$

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

2.4 Interpreting Images With Statistical Models

In order to interpret unseen images, we need to find the set of model parameters which best match the model to the image evidence. This set of model parameters then describes the object's shape and/or appearance (depending on the nature of the model used), and can therefore be used to further interpret the image, for instance to classify the object.

In this section we describe two algorithms which aim to locate the model parameters that best represent the target object in a new image.

A set of model parameters, $\boldsymbol{c}$, can be thought of as an *hypothesis* for the true location of the object. In general the methods in this section define a fit function, $F(\boldsymbol{c})$, which evaluates hypothesis $\boldsymbol{c}$ against an image, returning small values for accurate hypothesis. A way of optimising the value of $F(\boldsymbol{c})$ is also an important part of the algorithm.

2.4.1 Active Shape Models

The Active Shape Model algorithm (ASM) provides a method of fitting a PDM to an image in order to get a description of the object's shape. Many other methods for locating shape information exist [80, 41], but ASMs are widely used. The remainder of this section will provide a brief overview of the technique. For a more in-depth discussion see [23].

Modeling Local Image Structure

The ASM combines the shape constraints implied by a PDM with a local search scheme which searches for each of the landmarks independently. During a training phase, a grey level profile model is built for each landmark.

For every landmark in each training image a grey level profile can be sampled in a direction perpendicular to the boundary on which the landmark lies (See Figure 2.5). After normalising the profiles we can calculate the mean, $\bar{\boldsymbol{g}}$, and the covariance, $\boldsymbol{S}$ for each landmark.

The quality of fit of a new sample profile, $\boldsymbol{g}_s$, is given by:

$$f(\boldsymbol{g}_s) = (\boldsymbol{g}_s - \bar{\boldsymbol{g}})^T \boldsymbol{S}^{-1} (\boldsymbol{g}_s - \bar{\boldsymbol{g}}) \tag{2.15}$$

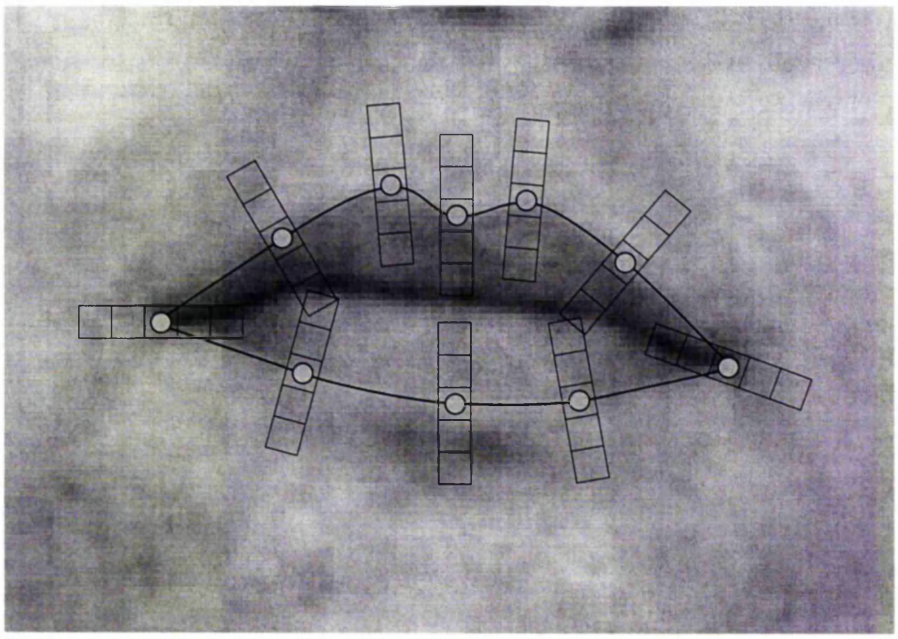


Figure 2.5: Grey-levels sampled around each landmark.

This is a Mahalanobis distance, thus small values imply a good match. This quality of fit measure is used during search to locate desirable positions towards which the landmarks should be encouraged to move.

Searching

Given an initial hypothesis for the approximate locations of the model points in the image frame, the following search scheme can be used to refine the hypothesis to fit the image evidence.

By choosing a set of shape parameters, $\boldsymbol{b}$, we can define the shape of a model instance, $\boldsymbol{x} = \bar{\boldsymbol{x}} + \boldsymbol{Pb}_s$, in the model frame. An instance, $\boldsymbol{X}$, of the model is created in the image frame by defining the position, orientation and scale:

$$\boldsymbol{X} = T_{x_t,y_t,\theta,s}(\boldsymbol{x}) = T_{x_t,y_t,\theta,s}(\bar{\boldsymbol{x}} + \boldsymbol{Pb}_s) \qquad (2.16)$$

where $T_{x_t,y_t,\theta,s}$ performs a scaling by s, a rotation by θ and a translation by (x_t, y_t). The initial approximation is formed by choosing values for $\boldsymbol{b}_s$, s, θ and (x_t, y_t). An iterative scheme is used to repeatedly improve this initial hypothesis. The grey level models and Equation 2.15 are used to search around the current location of each landmark, $\boldsymbol{X}$, seeking a better match. The search is in a direction normal to the boundary on which the landmark is suppose to lie (Figure 2.6). The best position for each landmark is placed in a adjusted shape vector, $\boldsymbol{X}'$,

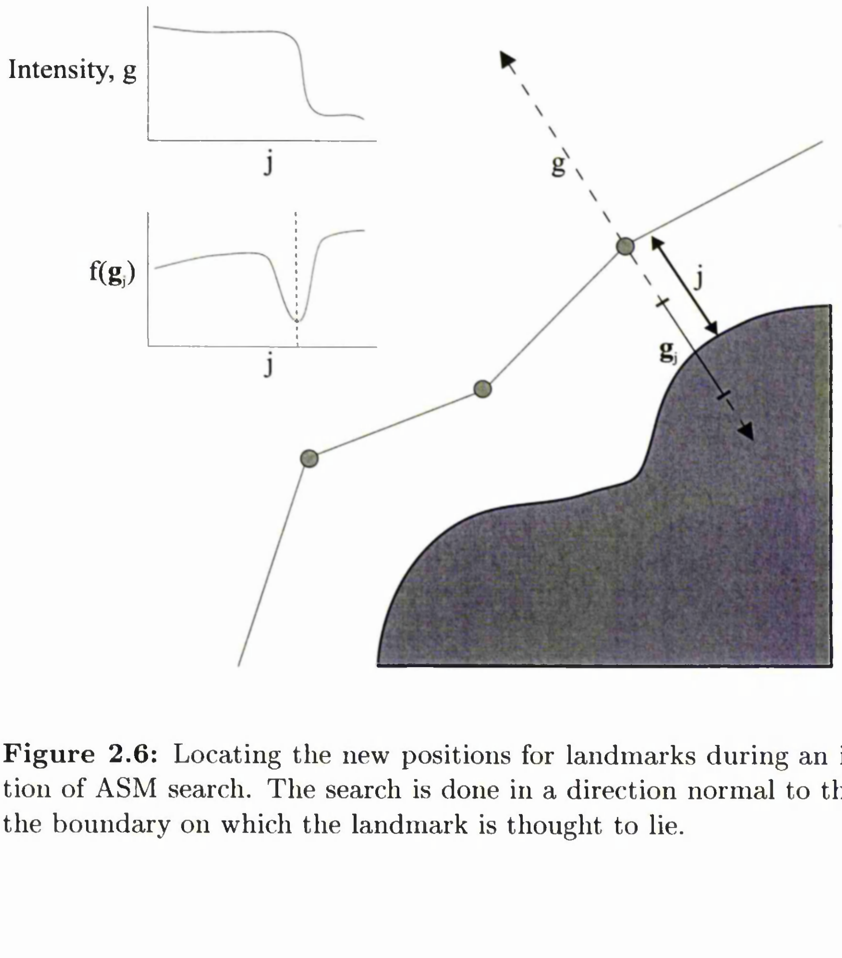


Figure 2.6: Locating the new positions for landmarks during an iteration of ASM search. The search is done in a direction normal to that of the boundary on which the landmark is thought to lie.

$$\boldsymbol{X}' = (x'_1, x'_2, \ldots, x'_n, y'_1, y'_2, \ldots, y'_n) \tag{2.17}$$

where (x'_i, y'_i) is the desired position of landmark i.

The adjusted shape vector, $\boldsymbol{X}'$, is used to update the model in two separate stages. First a new pose, $(x'_t, y'_t, s', \theta')$, is chosen to move the current model points, $\boldsymbol{x}$, as near as possible the their desired locations, $\boldsymbol{X}'$. The remaining differences between the model points and their desired locations in the model frame are known as the residual displacements, $\delta\boldsymbol{x}'$. Secondly, the model parameters, $\boldsymbol{b}$, are adjusted to minimise the residual displacements. It can be shown [25] that the optimum adjustments, $\delta\boldsymbol{b}$, to the models parameters, $\boldsymbol{b}$, are given by:

$$\delta\boldsymbol{b} = \boldsymbol{P}^T \delta\boldsymbol{x}' \tag{2.18}$$

This has the effect of minimising the residual displacements. By applying limits (for instance ± 3 s.d.) to $\boldsymbol{b}$ we ensure that the shape of the landmarks remains legal. Further iterations are calculated until the model parameters no longer change. At this point the algorithm has converged and if successful, the pose and model parameters describe the location of a solution. Figure 2.7 [†] shows an example of a successful ASM search.

The algorithm is best applied in a multi-resolution search framework, which leads to a faster and more reliable search [26, 19].

2.4.2 Active Appearance Models

Active Appearance Models (AAM) provide a way of fitting Combined Appearance Models directly to an image. They differ from Active Shape Models by taking advantage of all the available image evidence, not just the shape constraints and local landmark profiles. The AAM was first introduced by Edwards, Cootes and Taylor [32]. Cootes *et al* [17] provide a more detailed account.

[†]This illustration was kindly provided by Dr. G Edwards

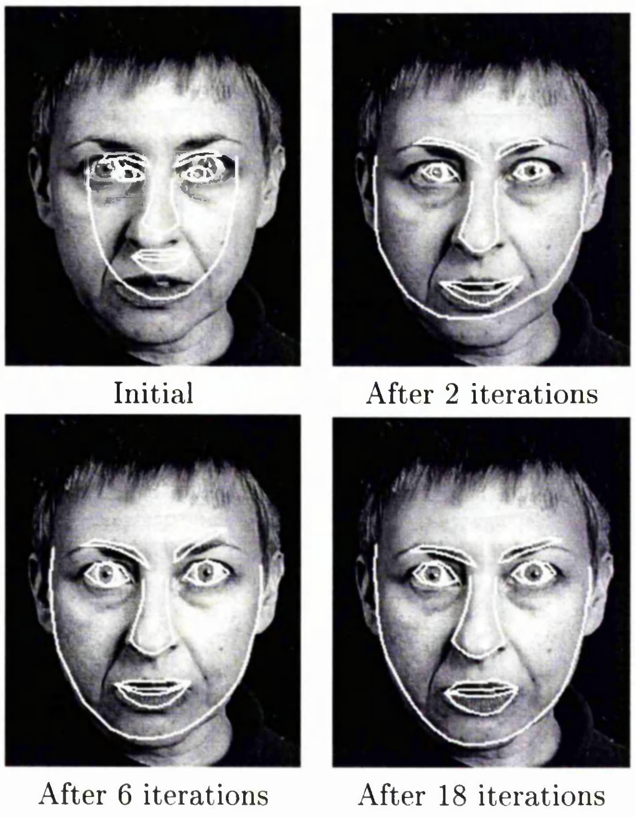


Figure 2.7: Locating a face using the Active Shape Model search algorithm.

Assuming a reasonably good starting position, fitting the AAM to a new image can be seen as an optimisation problem where we seek to minimise the difference between the image and the synthesised patch produced by the AAM. This difference can be represented by a vector $\delta \boldsymbol{g}$. $\delta \boldsymbol{g}$ is defined as:

$$\delta \boldsymbol{g} = \boldsymbol{g}_i - \boldsymbol{g}_m \tag{2.19}$$

where $\boldsymbol{g}_m$ is a vector of grey levels sampled from the synthesised patch and $\boldsymbol{g}_i$ is a corresponding vector of grey levels sampled from the image which lies directly under the current synthesised path.

Combined models of appearance are specific and will therefore not generate illegal instances. If we can reduce $|\delta \boldsymbol{g}|^2$ sufficiently we would ensure that the image, $\boldsymbol{g}_i$, represents something similar to the model $\boldsymbol{g}_m$. $|\delta \boldsymbol{g}|^2$ can be minimised by varying the model parameters $\boldsymbol{c}$ which we will assume for simplicity also contain scale, in-plane orientation and translation.

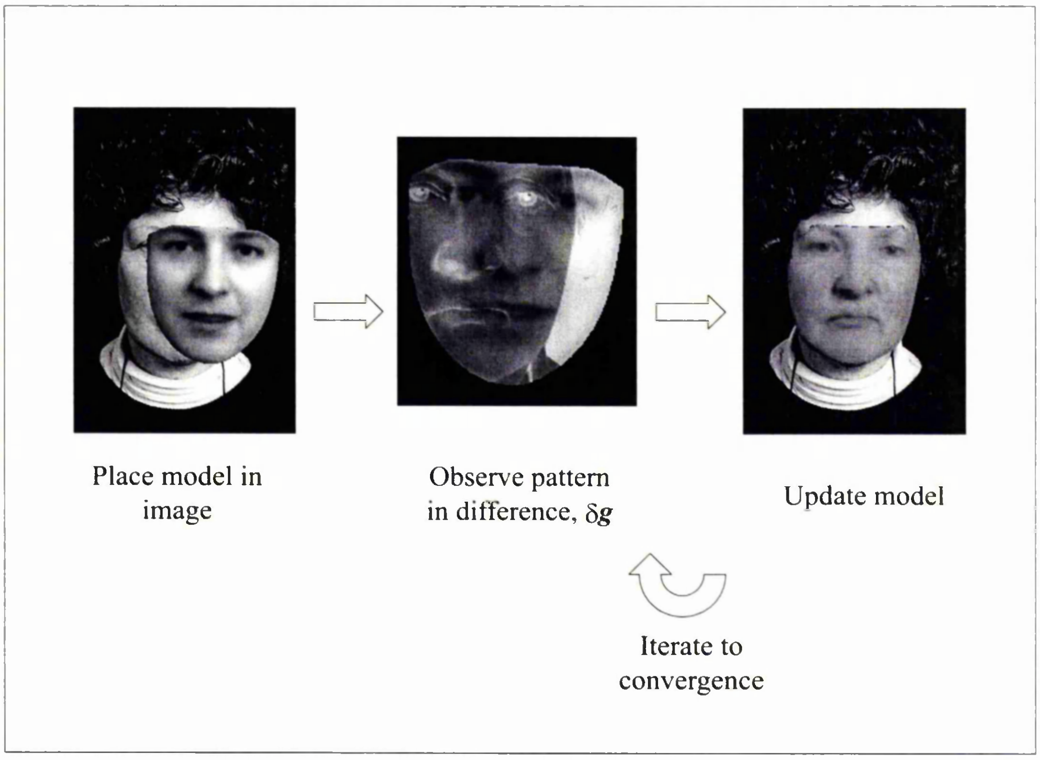


Figure 2.8: Overview of AAM search scheme.

The number of model parameters, $\boldsymbol{c}$, is often sufficiently large that using a standard optimisation algorithm would be inefficient. The AAM algorithm takes advantage of the fact that each attempt to match the model to a new image is a similar optimisation problem and it is possible to learn something about it in advance. In particular there is a strong relationship between the spatial pattern of the errors $\delta \boldsymbol{g}$ and parameter displacement from the true solution to the search $\delta \boldsymbol{c}$.

The relationship between $\delta \boldsymbol{g}$ and the error in the model parameters, $\delta \boldsymbol{c}$, is assumed to be linear:

$$\delta \boldsymbol{c} = \boldsymbol{A}\delta \boldsymbol{g} \tag{2.20}$$

$\boldsymbol{A}$ is found by performing multivariate linear regression [47] on a number of known model displacements, $\delta \boldsymbol{c}$, and the resulting difference images, $\delta \boldsymbol{g}$.

Given this relationship we can predict a set of model displacements from a difference image. This is the heart of the following iterative search procedure:

- evaluate the error vector $\delta \boldsymbol{g} = \boldsymbol{g}_s - \boldsymbol{g}_m$
- evaluate the current error $E = |\delta \boldsymbol{g}|^2$
- Compute the predicted displacement, $\delta \boldsymbol{c} = \boldsymbol{A}\delta \boldsymbol{g}$
- set $k = 1$
- let $\boldsymbol{c}' = \boldsymbol{c} - k\delta \boldsymbol{c}$
- sample the image at this new prediction, and calculate a new error vector, $\delta \boldsymbol{g}'$
- if $|\delta \boldsymbol{g}'|^2 < E$ then accept the new estimate, $\boldsymbol{c}'$,
- otherwise try at $k = 0.5$, $k = 0.25$ etc.

This procedure is repeated until no improvement is made to the error, $|\delta \boldsymbol{g}|^2$, and convergence is declared. Figure 2.8 provide an overview of this scheme and Figure 2.9 illustrates some search examples.

2.5 Related modeling techniques

Kass *et al* [51] introduced Active Contour Models (or 'snakes') which are energy minimising curves. The energy of the snake has an internal term and an external

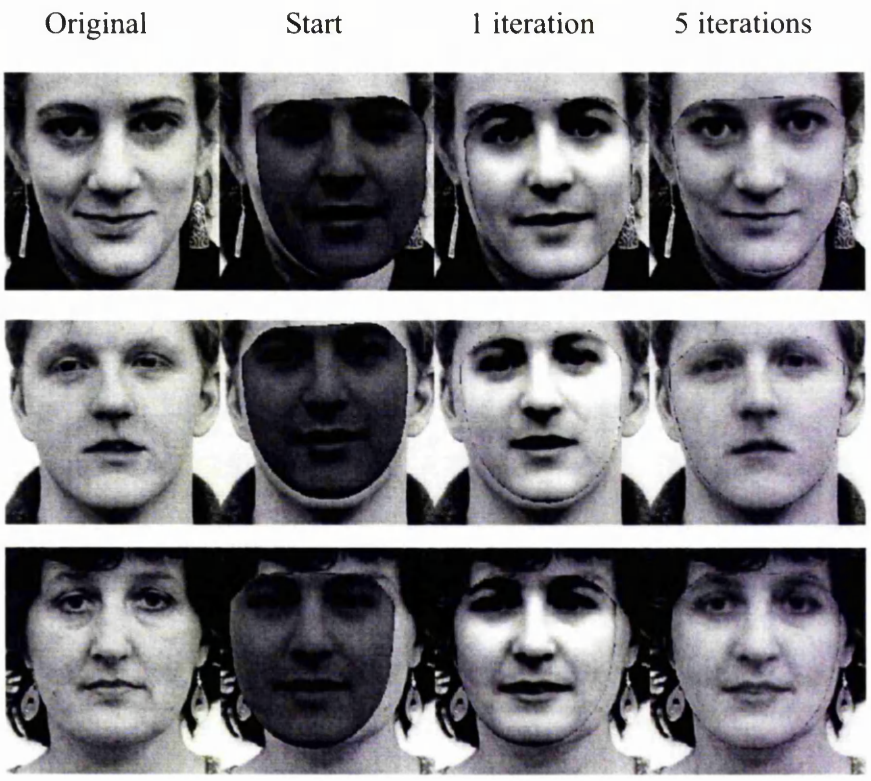


Figure 2.9: Examples of AAM search. Original image on left. Iterations 1,2,5 shown on right.

term. The internal term aims to impose smoothness on the curve and the external term encourages movement towards image features. They are a useful tool for locating general outlines, but since no model (other than smoothness) is included they are not the best solution for locating objects of a known shape.

Lades *et al* [55] describe Elastic Graph Matching. Elastic Graph Matching has been used for classifying hand gestures [80] and face recognition [94]. Objects are represented as labeled graphs, where the nodes represent local image information (the response of Gabor based filters). When a new image is presented to the system, the graph is overlaid and allowed to deform in order to minimise an energy function. The energy function is based on the graph deformation and the response of the Gabor filters at the new node positions. However they don't impose strong shape constraints

Staib and Duncan [76] represent shapes using fourier descriptors of closed curves. The choice of coefficients affects the curve complexity. Point distribution models as described in Section 2.1 are much more general, for instance fourier models can only describe closed boundaries.

Sclaroff and Isidoro describe Active Blobs [72]. The approach is broadly similar to the Active Appearance Models, in that they learn the relationship between image error and parameter offset in a training phase. The main difference is that Active Blobs are derived from a single training image, whereas Active Appearance Models are derived from a number of training images.

A number of authors propose physically based models. That is to allow a prototype to vary according to some physical model. Bajcsy and Kovacic [4] describe a volume model of the brain that deforms elastically to generate new examples. Christensen *et al* [15] describe a viscous flow model of deformation which they also apply to the brain. The resulting algorithm is very computationally expensive. The modes of variation learnt from a training set are likely to be more appropriate that those of a physically based model.

2.6 Summary

In this chapter we have motivated the need for techniques which can automatically or semi automatically provide a consistent set of landmarks across a set of images. We have described a number of modeling techniques, which are reliant upon a labeled set of training images and also a number of search schemes, which allow the models to be used directly for image interpretation.

The techniques presented in this chapter, particularly the Active Shape Models have proved extremely popular in a wide and diverse range of applications such as medical [22, 23, 21, 74, 75, 16], face recognition [35, 32], animal behavior [77] and industrial inspection.

Because of the success of these techniques research is being carried out in order to further improve their robustness [21, 20, 18, 33, 71]. Although much has already been achieved in this area, the sheer amount of new research indicates that we are yet to see the full potential of these modeling techniques.

Chapter 3

Automatic model building: A background

This chapter will review a number of schemes, previously published, which attempt to locate correspondences automatically. Relatively few authors consider the problem of finding consistent correspondences across a training set of images without the use of some prior model. Other relevant work attempts to find a transformation between a pair of images. It is possible that some of these schemes could be adapted for multiple images.

The papers reviewed in this chapter are categorised under three headings:

- *Pair-wise* schemes attempt to locate a transformation between pairs of images.
- *Serial* schemes process the training images in a serial manner (one after the other), generating a consistent transform across multiple images.
- *Parallel* schemes optimise a correspondence across multiple images in a parallel fashion.

3.1 Pair-wise schemes

Algorithms which fall into this category are often known as image registration techniques. Image registration requires finding an optimal transformation between an image pair, the *source* and *target*. We review a number of image registration techniques that are of relevance to this thesis.

Christensen [14] describes a technique for locating consistent transformations between a pair of images. Christensen notes that a fundamental problem with a number of pair-wise image registration techniques is that the estimated transform from image A to B is not equal to the inverse of the transformation estimated from image B to A. This inconsistency means that the correspondence between images A and B is ambiguous. Rather than estimate the transformations from A to B and from B to A separately, they are estimated jointly whilst enforcing a consistency constraint which ensures that the transformations are symmetric. If $h(x)$ is the transformation from image A to B and $g(x)$ is the transformation from image B to A, then Christensen attempts to calculate the transformations g and h by minimising a cost function that is a function of $B(g(x)) - A(x)$ and $A(h(x)) - B(x)$.

Lester *et al* [58] describe a non-linear registration algorithm which allows for different types of viscous fluid model. They use prior knowledge of the object class to specify a number of *inhomogeneity* paradigms to help further constrain the problem. For each image region a fairly complex prior description of its properties is required. This includes information such as the importance of the region, if it is strongly or weakly deformable and also the deformation model (affine etc) of the region. Lester shows that this extra information helps create robust registration. Lester manually specifies the region information, an automatic system might attempt to determine some of this information from the training set.

Hill *et al* [43, 44] present a method of corresponding the boundaries of two shapes. The algorithm approximates one of the boundaries by a sparse polygonal approximation. The polygonal approximation is found by using an iterative scheme to select points with a high *critical value.* These points tend to lie on the apex of regions of high curvature. A matching sparse polygon is sought on the second boundary. It should have a similar shape and representation error to the first. This is achieved by optimising a cost function using a greedy algorithm.

3.2 Serial schemes

Serial schemes tend to start with a few training images which are registered manually. These few images are used to train a model. Further images are added to the model by first calculating a model to image mapping (i.e. putting the new training image in correspondence with the model). With only a few training examples in the model, the model often struggles to generalise to new training images. In this section we discuss strategies used to overcome this problem.

3.2.1 Multidimensional Morphable Models

Jones and Poggio [48] describe flexible models of the shape and texture of a certain object class, known as Multidimentsional Morphable Models. The model is a linear combination of the shapes and textures displayed in a set of training images, which for a given set of parameters can be used to render a new image. In order to build the model it is necessary for a dense correspondence between all training examples to be known. Vetter *et al* [84] present a bootstrapping technique for automatically determining this correspondence between training examples. The idea behind the bootstrapping algorithm is to start with a small morphable model consisting of two training images and then to add to its representational power by repeatedly adding new images by setting them in correspondence with the model. In order to set a new image in correspondence with the model, they first fit the inadequate model to the image giving a rough correspondence. The correspondence is then refined further using an optical flow algorithm. Finally the new training image is added to the model, increasing its generality.

A further improvement is to instead of incorporating one new training image at every iteration, is to first train a morphable model from the correspondences obtained from all training images using optical flow. Since optical flow results in noisy correspondence fields, the model is trained using only the most significant modes of texture and shape variation. At each iteration the current model is used to match to each training image using optical flow to refine the correspondence, giving a new improved correspondence field which can be used to retrain the model. At each iteration the model retrains more and more modes of texture and shape variation. This improvement turns the serial scheme into a parallel scheme.

Results demonstrate the algorithm on human faces which display identity variation, but no pose, lighting or expression variation.

3.2.2 Artifical Variation

In the previous section we described how Morphable Models add new variation to the model by first fitting the inadequate model to a new training in and then using a second technique (optical flow in this case) to capture the variation the model couldn't explain. This new variation is then added to the model. An alternative approach is to attempt to add some artificial variation to the model *before* searching a new image. Adding artificial variation improves the model's ability to generalise, allowing it to fit well to a training image it was previously unable to explain. This comes with a cost, if the artificial variation differs to that displayed in the object class, the model loses some of its specificity. This results in the model generating illegal instances.

A number of authors [66, 73, 50, 65] have proposed finite element methods. Finite element methods take a *single* instance of a shape and treat it as if it were made of a flexible material. The techniques of modal analysis give a set of linear deformations of the shape equivalent to the models of vibration of the original shape. However, the modes are somewhat arbitrary and don't represent the real variations which occur in a class of shapes.

Wang *et al* [93] combined artificial variation with statistical information obtain from a training set. This was achieved by adding extra variation (artificial variation) to the covariance matrix obtained from statistical analysis of the shapes displayed in a training set. The artificial variation is aimed to induce elastic modes of variation into the model. They showed that the statistical information improved an elastic model for non-rigid registration.

Cootes *et al* [24] extend the Active Shape Model algorithm (see Section 2.4.1) in order to improve its robustness when training on relatively few examples. They introduce artificial variation to the model which mimics elastic vibration modes. The approach generates these elastic modes when few training examples exist and changes smoothly to using more statistical modes of variation when the model is eventually trained on a large training set. Figure 3.1 shows some elastic modes applied to a neutral face.

3.3 Parallel schemes

Parallel schemes consider all the training images at once, and are not dependent on their ordering. Very few schemes consider training images in parallel because it often results in algorithms which are computationally complex.

Kotcheff *et al* [54] used direct optimisation to place the landmarks on a set of closed curves. They define a mathematical expression which measures the compactness and the specificity of a model. This gives a measure that is a function of the landmark positions on the training set of curves. A genetic algorithm is used to ad-

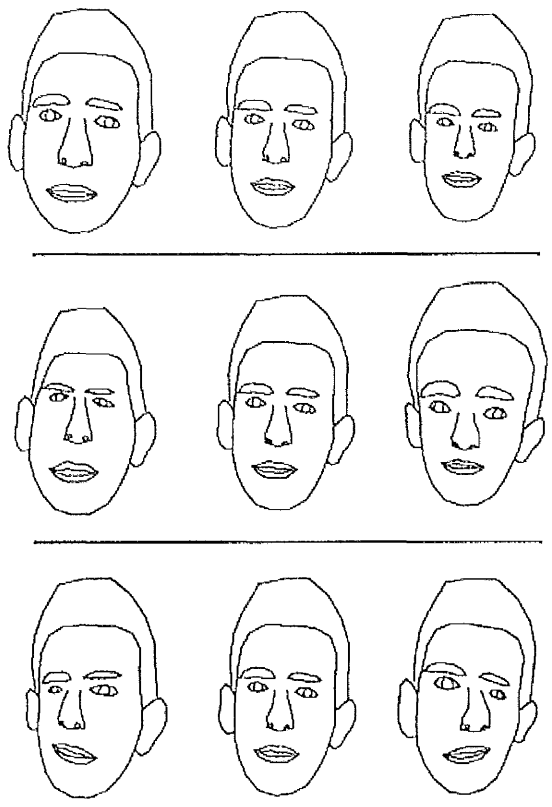


Figure 3.1: Elastic modes of variation applied to a neutral face.

just the point positions so as to optimise this measure. The approach seems promising when applied to boundaries of a number of object classes such as hands, resistors and heart ventricles. But it is not obvious how it could be extended to full 2D (region based) or 3D problems and still be tractable to compute.

Hill and Taylor [42] describe a parallel scheme for the automatic landmark generation for Point Distribution Models. Landmarks are generated for a training set of pixellated 2D boundaries. The algorithm is a two-stage process in which a pair-wise corresponder is first used to establish an approximate set of landmarks on each of the example boundaries. The first stage can be represented as a binary tree as shown in Figure 3.2. The leaves are the original training set and each node is a pair-wise corresponder which computes the mean shape from its two child nodes. The mean shape of the entire training set is found at the root. A set of landmarks is placed on this mean shape and then propagated back along the branches to the tree leaves giving approximate landmarks for each training shape. In the second phase the landmarks are refined using an iterative non-linear optimisation scheme. This is achieved by attempting to reduce the total absolute variance in the model whilst trying to explain as much of the variance using as few modes of variation as possible.

Brett *et al* [11]describe a framework similar to that of Hill's[42] for 3D automatic landmark generation. They construct a binary tree of merged shapes as shown in Figure 3.2. A polyhedral-based pair-wise corresponder is used to merge the shapes. The approach is used to produce a set of landmarks upon examples of the left brain ventricle.

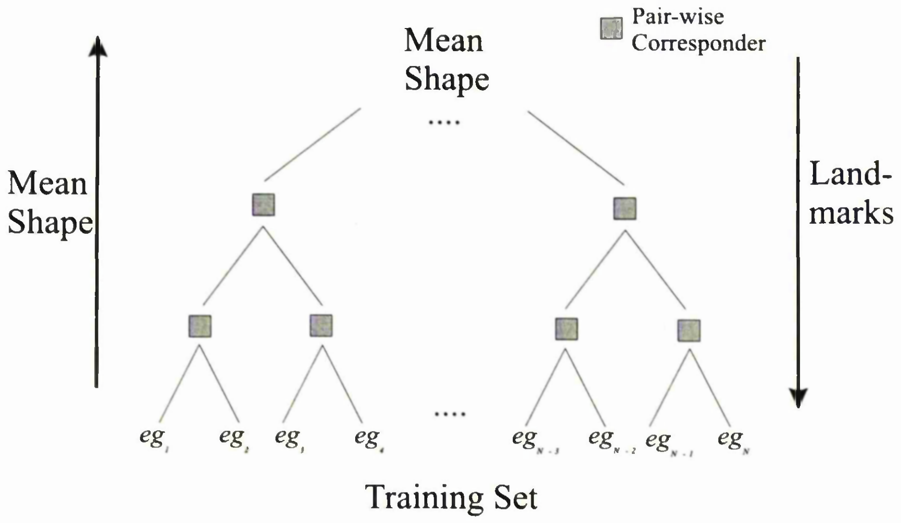


Figure 3.2: Generating the Mean Shape and Approximate Landmarks

3.4 Summary

We have reviewed several methods for computing a correspondence between 2 or more training images. The work which provides the closest answer to the aims of this thesis is that of Hill [42] and Jones's Multidimentional Morphable Models. As already mentioned Hill's work is applied only to the boundaries of a shape. We also require landmarks to represent the object's internal deformations.

Morphable models require a full dense correspondence between all training examples and hence the use of optical flow to compute the correspondence. For Active Shape Models and Active Appearance Models only a partial correspondence is necessary which is provided by landmarks. In this thesis we will explore feature based methods from automatic landmarking.

Chapter 4

Local Feature Descriptors

4.1 Introduction

This chapter reviews the various types of Local Feature Descriptors that are commonly described in the literature and how they can be used to measure the similarity between various image regions. Local Feature Descriptors allow the local image structure around a image pixel to be quantified in some form. Typically, applying a Local Feature Descriptor to an image will produce a *feature vector*, $\boldsymbol{v_x}$, for each point, $\boldsymbol{x}$, in the image. Some feature descriptors can be applied at various scales, resulting in a feature vector, $\boldsymbol{v_{x,\sigma}}$, for each scale, σ, at each point, $\boldsymbol{x}$, in the image.

Feature descriptors are typically used in vision applications at the lowest level to assess if two image regions have similar characteristics. The higher level architecture uses this information together with other information and constraints (possibly from a pre-trained model) to interpret an image in some way. Triesch *et al* used feature descriptors based on Gabor Wavelets to construct a model of the human hand and then to match the model to unseen images of hands. The feature descriptors are used in the elastic graph matching which allows the hand model to fit to an unseen image. Several authors have used feature descriptors to form models of the human face. These models have been used for a variety of tasks such as coding facial expressions [59], facial expression recognition [45], facial identity recognition [49] and pose estimation [36].

4.2 Feature Coding

Feature coding is the process of constructing feature vectors. There are many different ways in which to code features, but they can generally be grouped into one of two classes. *Specific* coding methods attempt to enhance certain image structures such as corners, edges or blobs. *Non-specific* coding methods attempt to provide a way of describing any type of image structure, they are not biased to enhancing one particular type of feature.

4.2.1 Specific Feature Coding methods

Edge detectors

Edges or object boundaries have long been recognised as an extremely important part of the structure of an image. Often the shape of the outline of an object is sufficient to identify it. Experiments have also shown that edges are important to the human visual system [2].

Edges appear in images as intensity discontinuities. An edge can be described by two independent values; its magnitude and its direction. Given an image function, $f(\boldsymbol{x})$, the gradient magnitude, $s(\boldsymbol{x})$, and direction, $\phi(\boldsymbol{x})$, are given by [5]:

$$s(\boldsymbol{x}) = (\Delta_1^2 + \Delta_2^2)^{1/2}$$

$$\phi(x) = tan^{-1}(\Delta_2/\Delta_1)$$

where Δ_1 and Δ_2 are the *difference operators*, or a measure of the difference in grey levels in orthogonal directions. Numerous image operators have been suggested to calculate Δ_1 and Δ_2, Table 4.1 illustrates the three most popular [68, 69].

Method	Δ_1	Δ_2
Roberts	$\begin{array}{cc} 0 & 1 \\ -1 & 0 \end{array}$	$\begin{array}{cc} 1 & 0 \\ 0 & -1 \end{array}$
Prewitt	$\begin{array}{ccc} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{array}$	$\begin{array}{ccc} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{array}$
Sobel	$\begin{array}{ccc} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{array}$	$\begin{array}{ccc} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{array}$

Table 4.1: Difference operators

The difference operators in Table 4.1 are all approximately first order gaussian partial derivatives in orthogonal directions.

Corner Detectors

Corner detectors are another important feature detector. Corners refer to point features which are the loci of two-dimensional intensity change, i.e. second-order features. They occur at points of occlusion, at structural discontinuities and also at various curvature maxima.

Corner detectors have commonly been used to interpret scenes containing man made objects with physical corners, such as in buildings, although they have been applied to more natural scenes.

Several approaches to the problem of detecting corners have been reported and they can be broadly divided up into two groups:

The first group consist of first extracting edges as a chain code, and then searching for points of high curvature [29, 1] or fitting polygons to the chains and searching for line segment intersections [46]. In this case the feature vector, $\boldsymbol{v}$, would consist of the angle between the two edges which form the corner and the magnitude of their gradient.

The second group consists of approaches that work directly on a grey-level image. These techniques are based on the measurement of the gradients and the curvatures of the surface [92, 53, 30]. These measurements form the feature vector.

4.2.2 Non-specific Feature Coding Methods

Template matching

Template matching is one of the simplest methods of coding features. The feature vector is formed by sampling image grey levels from a grid placed directly over the image region to be coded. The scale of the feature can be set by changing the sampling rate between grid nodes. Figure 4.1 illustrates how template matching can be used to construct a feature vector describing the features within and around the eye.

Gabor Transform

The Gabor transform is used to reveal local spectral information in an image. Gabor kernels take the form of a complex plane wave restricted by an elliptical Gaussian envelope function as illustrated in Figure 4.2. A 2D Gabor Kernel, $\upsilon_{\boldsymbol{k}}$, is given by:

$$\boldsymbol{v}_{\boldsymbol{x},\sigma} = \begin{bmatrix} \boldsymbol{v}_1 \\ \boldsymbol{v}_2 \\ \cdot \\ \cdot \\ \cdot \\ \boldsymbol{v}_{24} \\ \boldsymbol{v}_{25} \end{bmatrix}$$

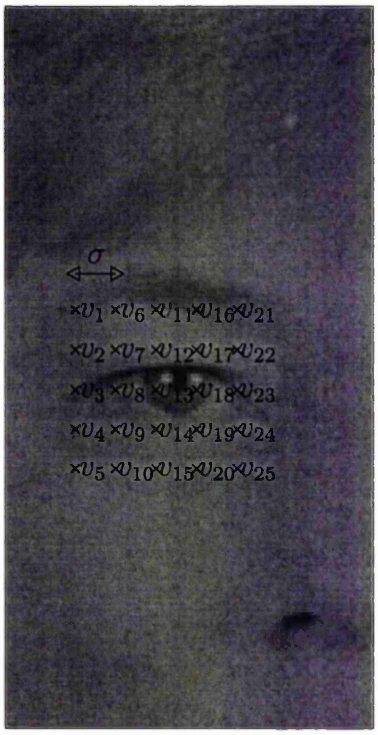


Figure 4.1: An illustration of how template matching is used to construct a feature vector. σ is the scale of the feature and $\boldsymbol{x}$ is the centre of the feature.

$$v_{\boldsymbol{k}}(\boldsymbol{x}) = \frac{\boldsymbol{k}^2}{\sigma^2} exp(-\frac{\boldsymbol{k}^2\boldsymbol{x}^2}{2\sigma^2})[exp(i\boldsymbol{k}\boldsymbol{x}) - exp(-\sigma^2/2)] \tag{4.1}$$

where $\boldsymbol{k}$ is complex and determines wavelength and orientation of the frequencies to be extracted and σ determines the ratio of window width to wavelength, i.e. the number of oscillations under the envelope function. This is further explained by Lee [57]

Lades *et al* [55] and Triesch *et al* [80] have both used Gabor-based filters in object recognition tasks. Because the response is dependent on the orientation of the filter, the convolution needs to be carried out at a number of angles, typically eight. This means that it is not ideal if speed is an important factor. Gabor filters are also used

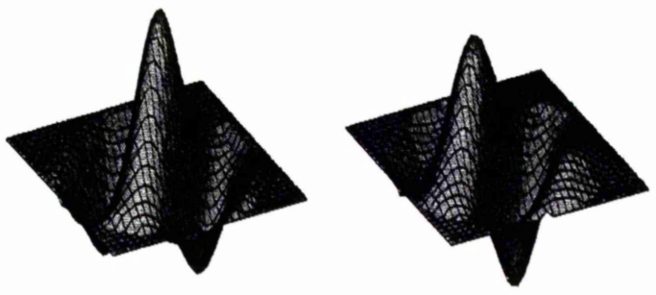


Figure 4.2: The real and imaginary parts of Gabor-based filters.

in texture analysis [10, 37, 60].

Gaussian Partial Derivatives

Gaussian Partial Derivatives are a set of operators which describe the differential structure of an image region (e.g. about a feature) in a way which is dependent on the chosen Cartesian coordinate system. This is done using a scaled set of differential operators. The zeroth order operator, on which the rest of the operators are based, is the normalised isotropic Gaussian. The higher order operators are generated from derivatives of the Gaussian, allowing one to study the differential structure of the image up to any order using a set of discrete convolution filters.

The normalised Gaussian Kernel is given by:

$$G(\boldsymbol{x};\sigma) = \frac{1}{(2\pi\sigma^2)^{\frac{D}{2}}} \cdot e^{(-\frac{|\boldsymbol{x}|^2}{2\sigma^2})} \tag{4.2}$$

where σ is the scale and D is the number of dimensions.

The first order Kernels are:

$$G_x(\boldsymbol{x};\sigma) = \tfrac{\partial}{\partial x}G(\boldsymbol{x};\sigma) \qquad G_y(\boldsymbol{x};\sigma) = \tfrac{\partial}{\partial y}G(\boldsymbol{x};\sigma)$$

The second order Kernels are:

$$G_{xx}(\boldsymbol{x};\sigma) = \tfrac{\partial^2}{\partial x^2}G(\boldsymbol{x};\sigma) \qquad G_{yy}(\boldsymbol{x};\sigma) = \tfrac{\partial^2}{\partial y^2}G(\boldsymbol{x};\sigma)$$

$$G_{xy}(\boldsymbol{x};\sigma) = \tfrac{\partial^2}{\partial xy}G(\boldsymbol{x};\sigma)$$

and so on.

The set of all such filters completely determine the local image structure at the given scale. Notice that each filter has a single parameter, σ, defining the scale of the filter.

Typically, researchers have tended only to use up to second or third order filters. This is because increasingly higher order becomes more and more susceptible to image noise. In addition, using further orders increases the dimensionality of the feature vector, which results in a computationally complex algorithm.

Figure 4.3 illustrates all the resulting Gaussian derivative filters up to and including the third order. Figures 4.4, 4.5 and 4.6 illustrate applying the Gaussian partial derivative kernels to an image at scales of 6, 4 and 2 standard deviations respectively. L_x is the result of convolving the image with G_x, L_y with G_y and so on. Convolving an image with a large scale filter results in a blurred image when compared with that of a small scale filter. Filters with a small scale reveal high frequency image structure and conversely filters with a large scale reveal low frequency image structure.

A feature vector, $\boldsymbol{v}_{x\sigma}$, based on Gaussian partial derivatives up to the second order would have the following form:

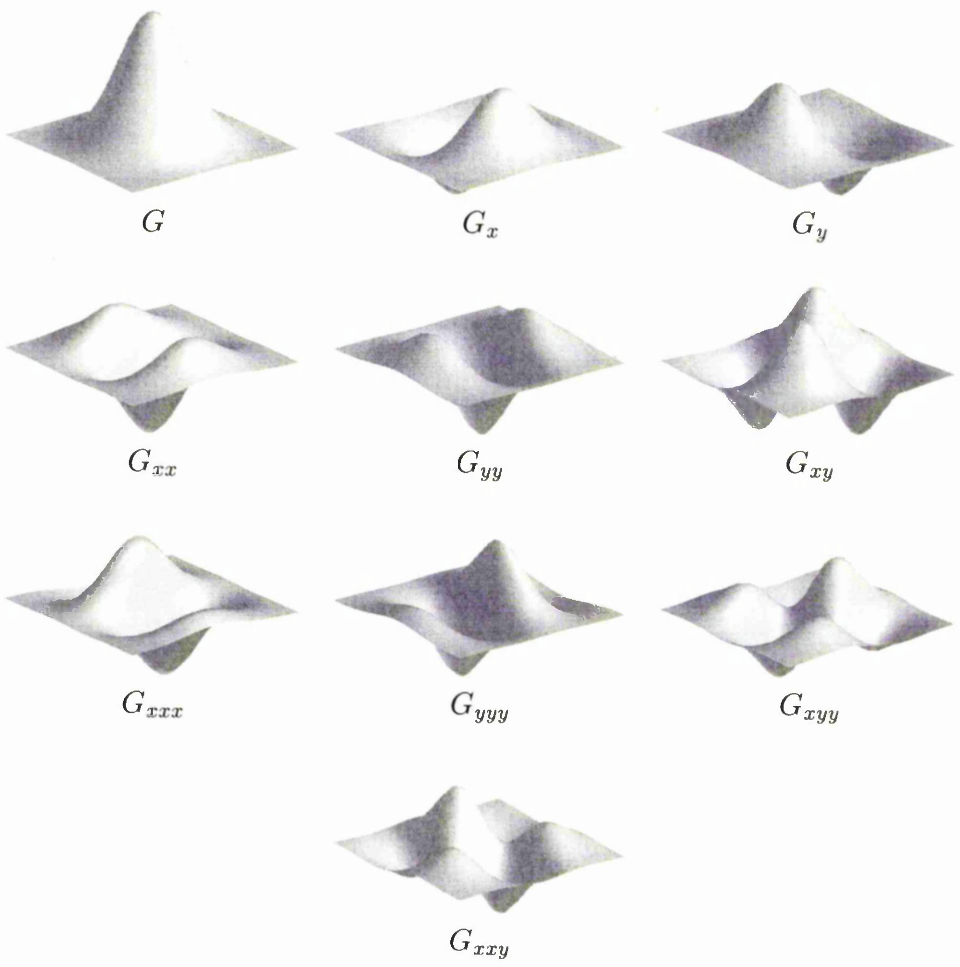


Figure 4.3: Gaussian partial derivative kernels up to and including the third order.

$$\boldsymbol{v}_{\boldsymbol{x},\sigma} = \begin{bmatrix} \boldsymbol{L}(\boldsymbol{x};\sigma) \\ \boldsymbol{L}_x(\boldsymbol{x};\sigma) \\ \boldsymbol{L}_y(\boldsymbol{x};\sigma) \\ \boldsymbol{L}_{xx}(\boldsymbol{x};\sigma) \\ \boldsymbol{L}_{yy}(\boldsymbol{x};\sigma) \\ \boldsymbol{L}_{xy}(\boldsymbol{x};\sigma) \end{bmatrix} \tag{4.3}$$

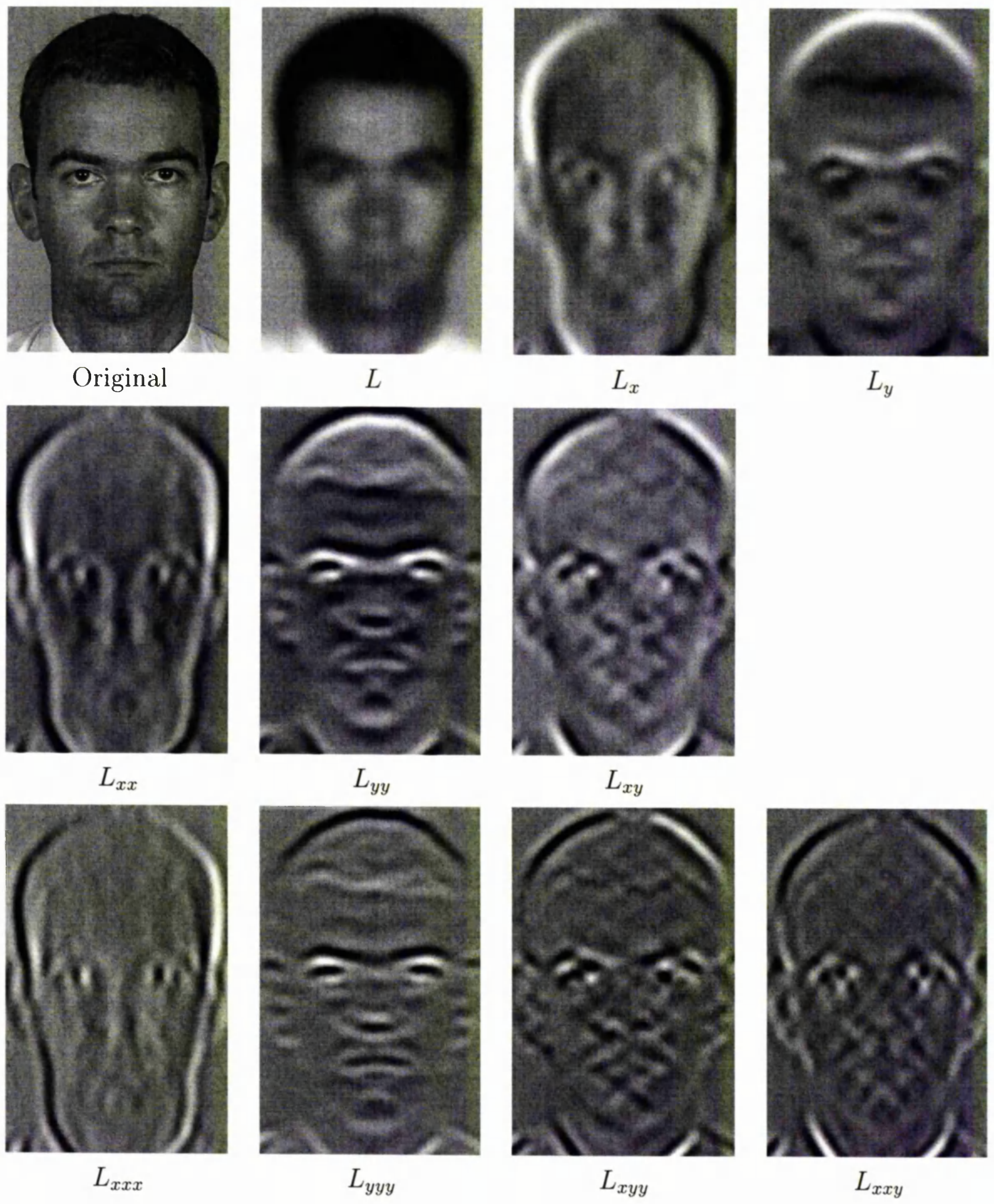


Figure 4.4: Gaussian partial derivative images up to the third order. $\sigma = 6$ pixels. The first row shows the 0^{th} and 1^{st} order, the second row the 2^{nd} order and the third row the 3^{rd} order. See text for details.

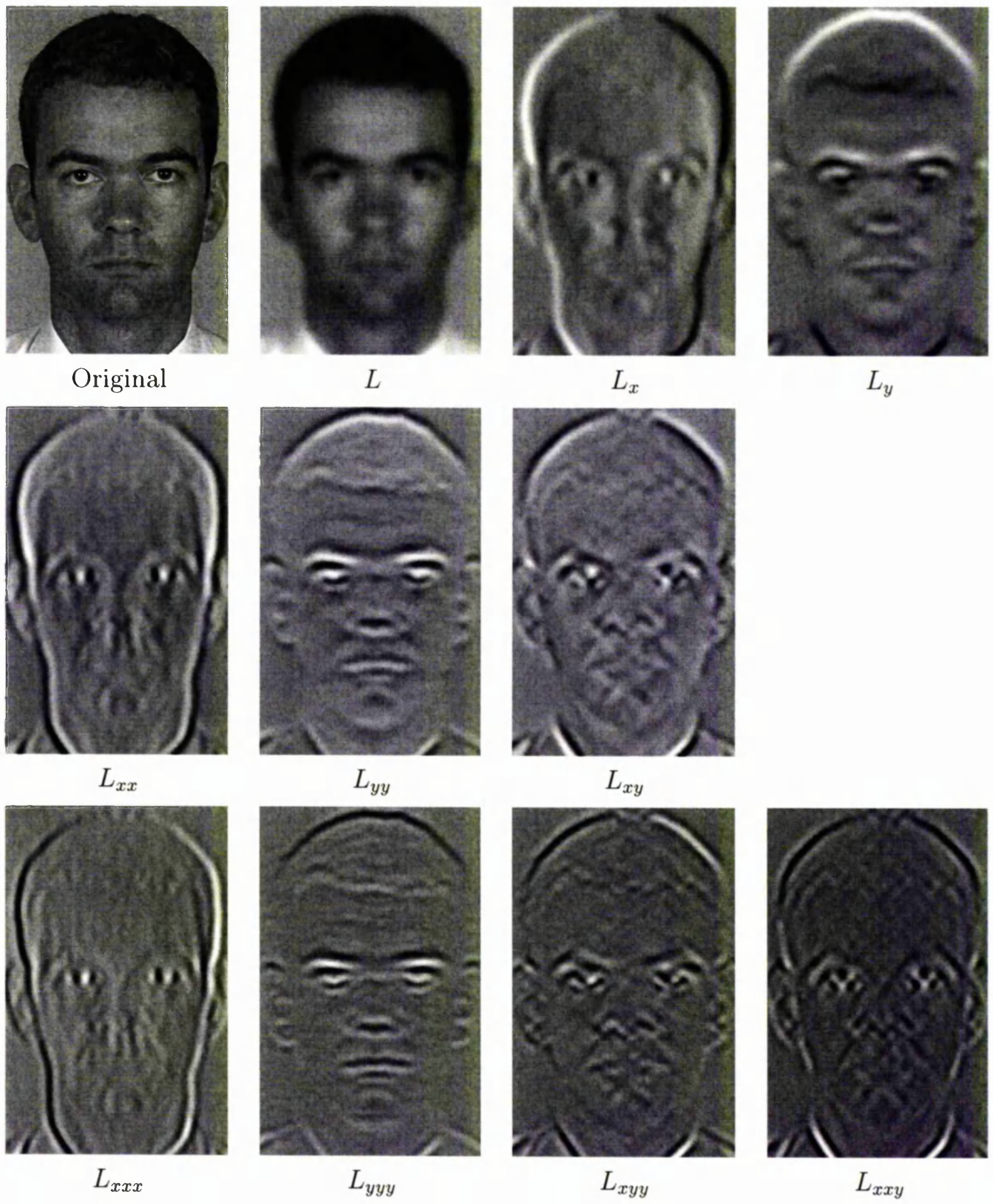


Figure 4.5: Gaussian partial derivative images up to the third order. $\sigma = 4$ pixels. The first row shows the 0^{th} and 1^{st} order, the second row the 2^{nd} order and the third row the 3^{rd} order. See text for details.

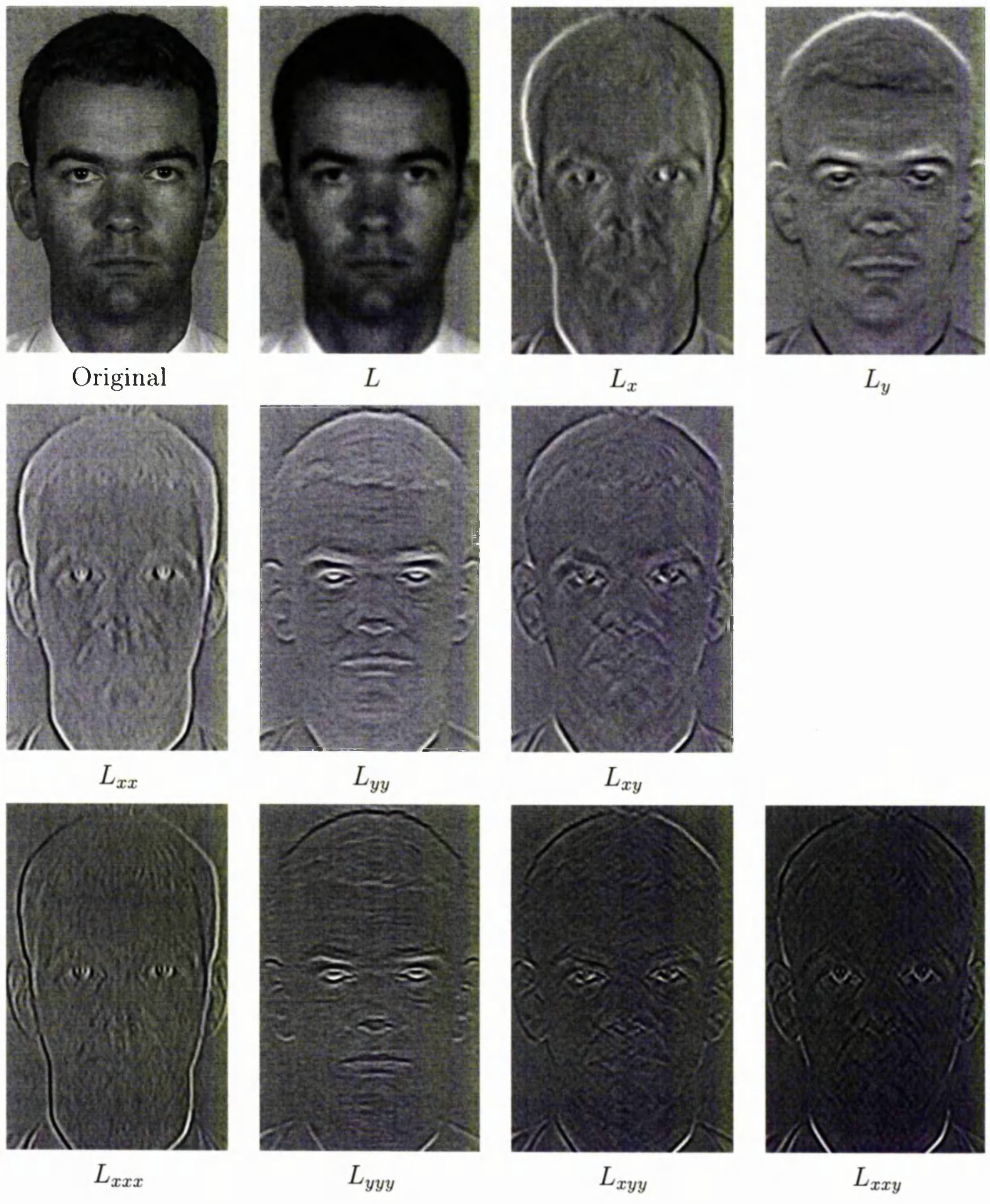


Figure 4.6: Gaussian partial derivative images up to the third order. $\sigma = 2$ pixels. The first row shows the 0^{th} and 1^{st} order, the second row the 2^{nd} order and the third row the 3^{rd} order. See text for details.

By dividing by $\boldsymbol{L}(\boldsymbol{x};\sigma)$ gives a feature vector which is invariant to image brightness,

$$\boldsymbol{v}_{\boldsymbol{x}\sigma} = \frac{1}{L(\boldsymbol{x};\sigma)} \begin{bmatrix} \boldsymbol{L}_x(\boldsymbol{x};\sigma) \\ \boldsymbol{L}_y(\boldsymbol{x};\sigma) \\ \boldsymbol{L}_{xx}(\boldsymbol{x};\sigma) \\ \boldsymbol{L}_{yy}(\boldsymbol{x};\sigma) \\ \boldsymbol{L}_{xy}(\boldsymbol{x};\sigma) \end{bmatrix} \tag{4.4}$$

Cartesian Differential Invariants

The response of Gaussian partial derivatives depends on the choice of Cartesian coordinate frame. However, certain combinations of filter response can be shown to be independent of the choice of coordinate frame. These combinations are known as *Cartesian differential invariants*. Table 4.2 shows a canonical set of eight two-dimensional polynomial invariants to third order, expressed in tensorial manifest invariant index notation [78].

Invariant	*Order*	*Manifest Invariant Notation*
I_1	1	L_iL_i
I_2	2	$L_iL_{ij}L_j$
I_3	2	$L_{ii}L_jL_j - L_{ij}L_iL_j$
I_4	2	$-\epsilon_{ij}L_{jk}L_iL_k$
I_5	3	$\epsilon_{ij}(L_{jkl}L_iL_kL_l - L_{jkk}L_iL_lL_l)$
I_6	3	$L_{iij}L_jL_kL_k - L_{ijk}L_iL_jL_k$
I_7	3	$-\epsilon_{ij}L_{jkl}L_iL_kL_l$
I_8	3	$L_{ijk}L_iL_jL_k$

Table 4.2: The set of 2D polynomial invariants to third order expressed in tensorial manifest invariant notation

Tensorial manifest invariant index notation is the most common notation used to express invariants. L_{ij} is the result of applying a second order Gaussian filter G_{ij} to an image, hence L_{ij} is a second order tensor. A polynomial invariant of the L terms can be constructed by expanding the Manifest Invariant notation using Einstein summation [52]. For example the second order invariant $L_iL_{ij}L_j$ expands as follows in two dimensions:

$$\begin{aligned} L_iL_{ij}L_j &= L_xL_{xx}L_x + L_xL_{xy}L_y + L_yL_{yx}L_x + L_yL_{yy}L_y \\ &= L_{xx}L_{x^2} + 2L_xL_{xy}L_y + L_{yy}L_{y^2} \end{aligned}$$

Each subscript, i, j etc is replaced with x, y and the result summed.

As well as the L tensors, ϵ_{ij} represents a constant tensor known as Epsilon or the Levi-Civita tensor. In D dimensions Epsilon has D indices and is defined as follows:

$$\epsilon_{i_1 \dots i_D} = \begin{cases} 1 & \text{if } (i_1 \dots i_D) \text{ is an even permutation of the natural order} \\ -1 & \text{if } (i_1 \dots i_D) \text{ is odd permutation of the natural order} \\ 0 & \text{otherwise} \end{cases}$$

Figures 4.7, 4.8 and 4.9 illustrate the responses of the Cartesian Differential Invariants at scales of 6, 4 and 2 standard deviations respectively. Convolving an image with a large scale filter results in a blurred image when compared with that of a small scale filter. Filters with a small scale reveal high frequency image structure and conversely filters with a large scale reveal low frequency image structure.

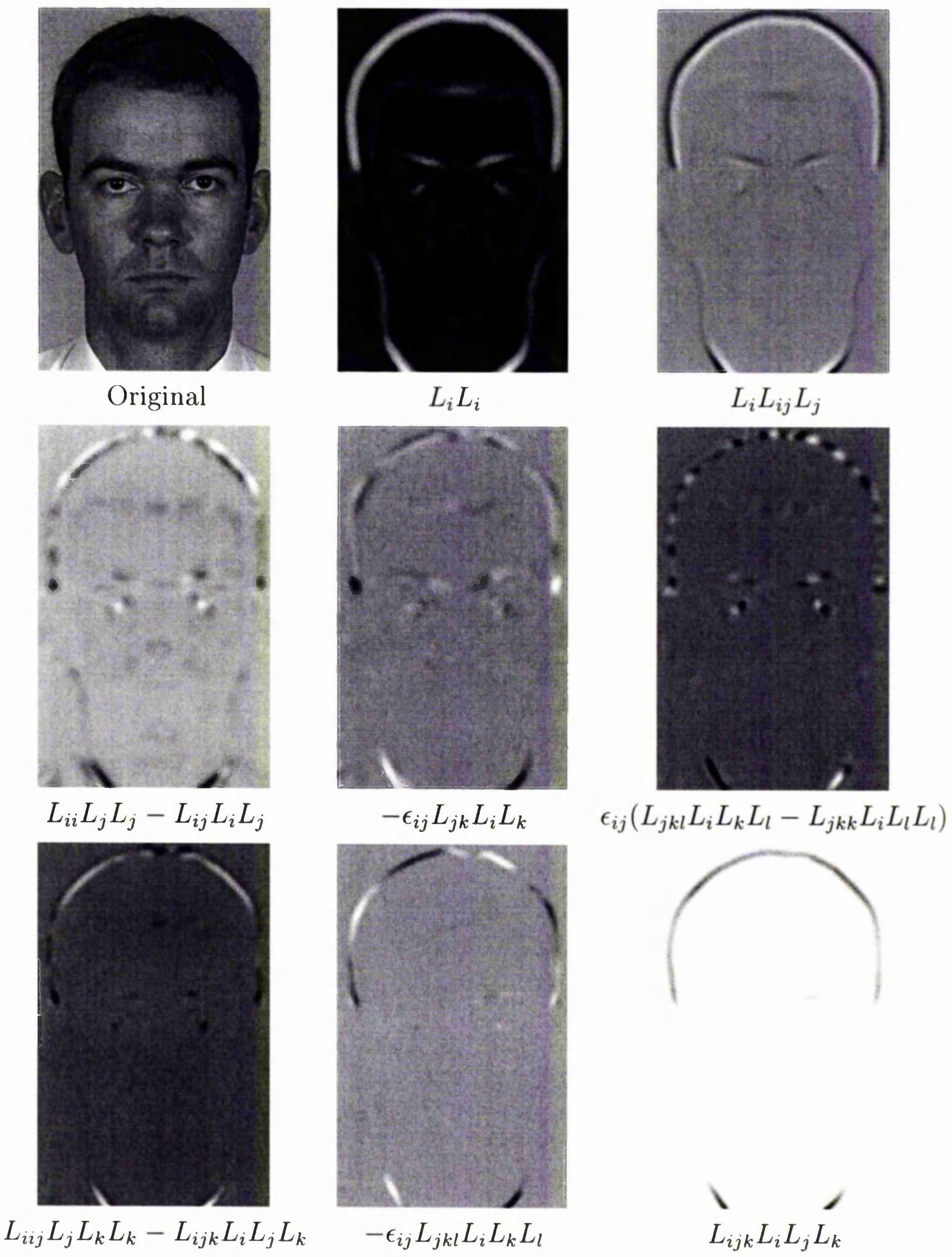

Original
$L_i L_i$
$L_i L_{ij} L_j$
$L_{ii} L_j L_j - L_{ij} L_i L_j$
$-\epsilon_{ij} L_{jk} L_i L_k$
$\epsilon_{ij}(L_{jkl} L_i L_k L_l - L_{jkk} L_i L_l L_l)$
$L_{iij} L_j L_k L_k - L_{ijk} L_i L_j L_k$
$-\epsilon_{ij} L_{jkl} L_i L_k L_l$
$L_{ijk} L_i L_j L_k$

Figure 4.7: Cartesian differential invariant images up to the third order. $\sigma = 6$ pixels. See text for details.

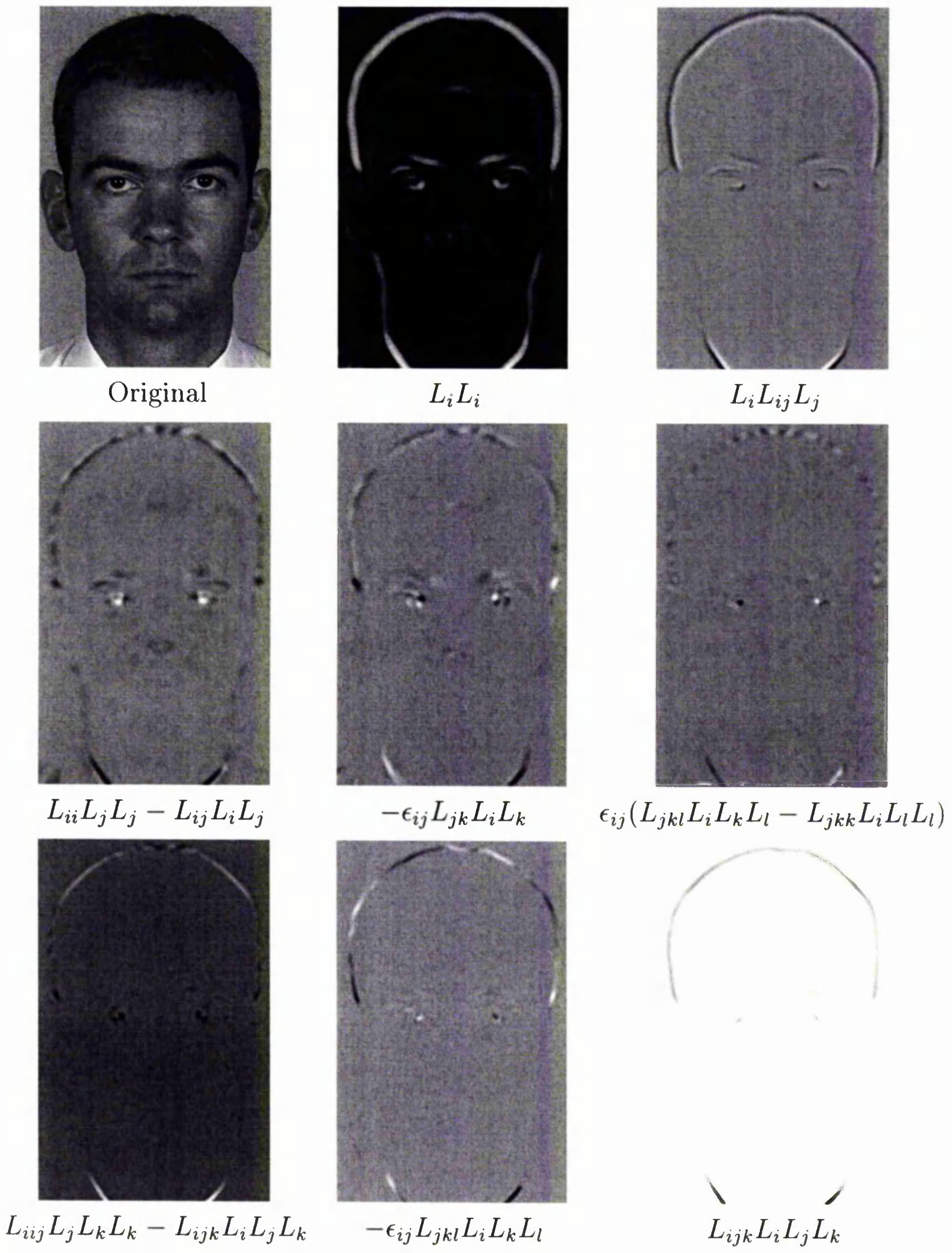

Original
L_iL_i
$L_iL_{ij}L_j$
$L_{ii}L_jL_j - L_{ij}L_iL_j$
$-\epsilon_{ij}L_{jk}L_iL_k$
$\epsilon_{ij}(L_{jkl}L_iL_kL_l - L_{jkk}L_iL_lL_l)$
$L_{iij}L_jL_kL_k - L_{ijk}L_iL_jL_k$
$-\epsilon_{ij}L_{jkl}L_iL_kL_l$
$L_{ijk}L_iL_jL_k$

Figure 4.8: Cartesian differential invariant images up to the third order. $\sigma = 4$ pixels. See text for details.

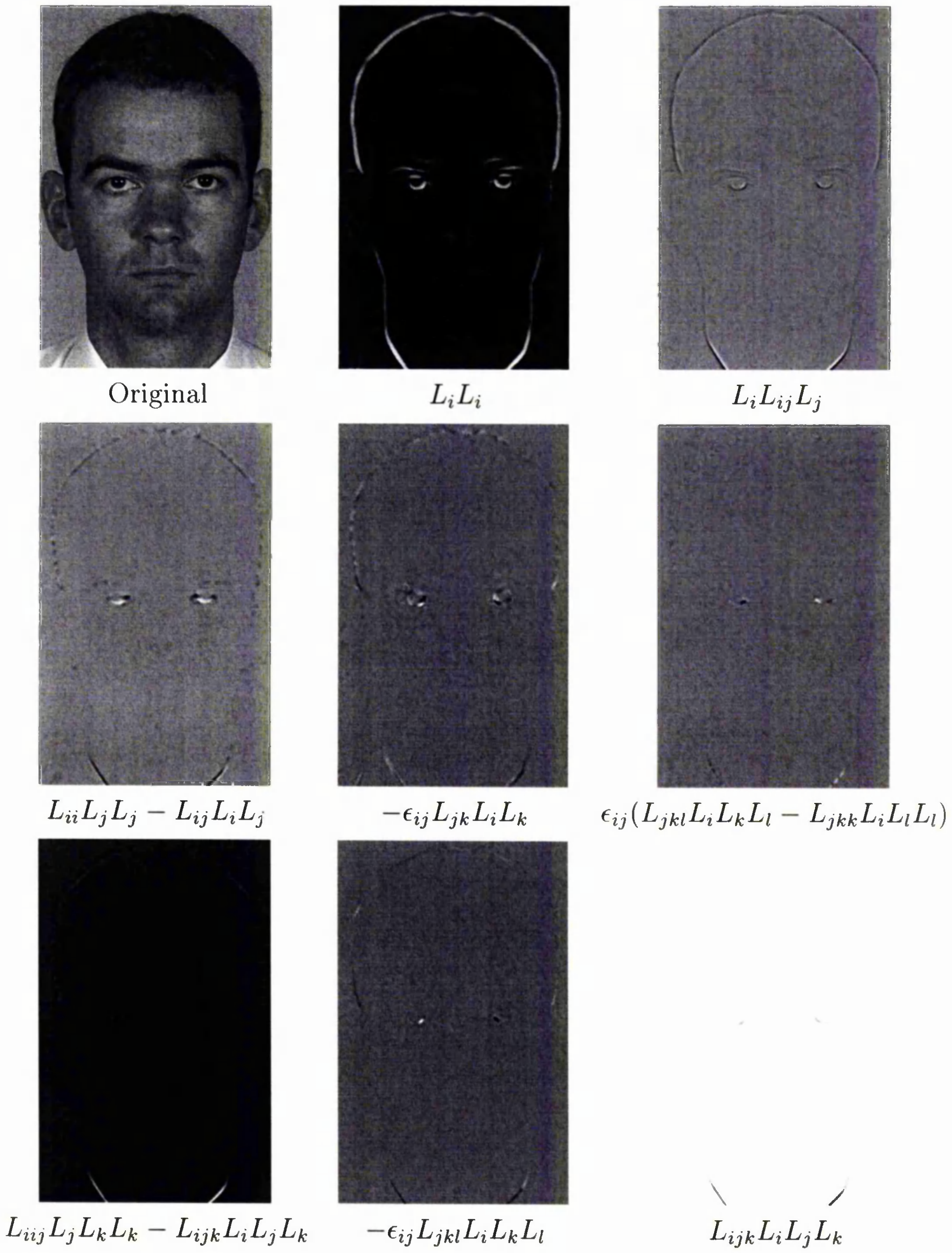

Original
L_iL_i
$L_iL_{ij}L_j$
$L_{ii}L_jL_j - L_{ij}L_iL_j$
$-\epsilon_{ij}L_{jk}L_iL_k$
$\epsilon_{ij}(L_{jkl}L_iL_kL_l - L_{jkk}L_iL_lL_l)$
$L_{iij}L_jL_kL_k - L_{ijk}L_iL_jL_k$
$-\epsilon_{ij}L_{jkl}L_iL_kL_l$
$L_{ijk}L_iL_jL_k$

Figure 4.9: Cartesian differential invariant images up to the third order. $\sigma = 2$ pixels. See text for details.

4.3 Measuring the Similarity Between two Features

In section 4.2 we covered some of the more popular ways in which features can be coded as a feature vector, $\boldsymbol{v}_{x\sigma}$. Here we discuss how to measure the similarity, $\delta_{\boldsymbol{v}_1\boldsymbol{v}_2}$, between two feature vectors, $\boldsymbol{v}_1$ and $\boldsymbol{v}_2$. This will allow us to compare features and to find good matches between features.

4.3.1 Angle

The angle between feature vectors is the simplest measure of similarity. A similarity measure based on angle is given by:

$$\delta_{\boldsymbol{v}_1\boldsymbol{v}_2} = \cos^{-1}\frac{\boldsymbol{v}_1 \cdot \boldsymbol{v}_2}{|\boldsymbol{v}_1||\boldsymbol{v}_2|} \tag{4.5}$$

The main disadvantage with this method is that the similarity of two feature vectors is only dependent on the relative direction. It assumes similarity is independent of the feature vector's magnitude. This is often not satisfactory as the magnitude does represent something meaningful in the image structure.

4.3.2 Squared Euclidean Distance

The Squared Euclidean distance (also known as *the sum of squares*) is given by:

$$\delta_{\boldsymbol{v}_1\boldsymbol{v}_2} = (\boldsymbol{v}_1 - \boldsymbol{v}_2) \cdot (\boldsymbol{v}_1 - \boldsymbol{v}_2) \tag{4.6}$$

Unlike Angle similarity only identical vectors will give $\delta_{\boldsymbol{v}_1\boldsymbol{v}_2} = 0$.

4.3.3 Mahalanobis Distance

The Mahalanobis distance is similar to the Euclidean Distance but allows extra information about the distribution from which $\boldsymbol{v}_1$ and $\boldsymbol{v}_2$ were drawn to be included. The Mahalanobis Distance can be used as a measure of relative probability under the assumption of a gaussian distribution of covariance $\boldsymbol{S}$. The Mahalanobis distance between $\boldsymbol{v}_1$ and $\boldsymbol{v}_2$ is given by:

$$\delta_{\boldsymbol{v}_1\boldsymbol{v}_2} = (\boldsymbol{v}_1 - \boldsymbol{v}_2)^T \boldsymbol{S}^{-1} (\boldsymbol{v}_1 - \boldsymbol{v}_2) \tag{4.7}$$

Figure 4.10 shows an example of where the Mahalanobis distance gives different results to the Euclidean distance.

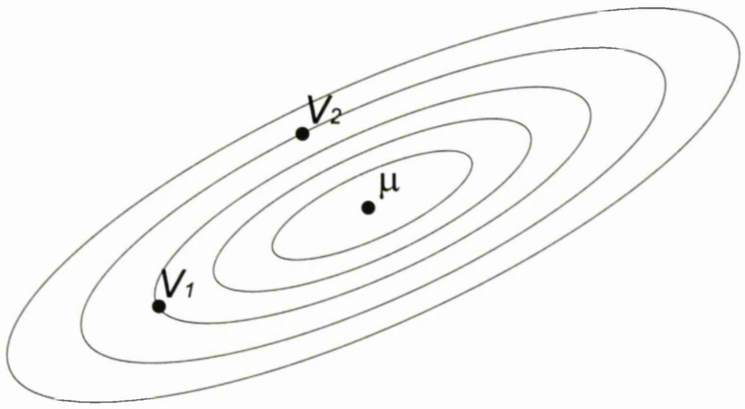


Figure 4.10: Illustration of the differences between Mahalanobis distances and Euclidean distances. The ellipses represent the distribution from which $\boldsymbol{v}_1$ and $\boldsymbol{v}_2$ were drawn. The Euclidean metric claims $\delta_{\boldsymbol{v}_1\boldsymbol{\mu}} > \delta_{\boldsymbol{v}_2\boldsymbol{\mu}}$, where as the Mahalanobis distance claims $\delta_{\boldsymbol{v}_1\boldsymbol{\mu}} < \delta_{\boldsymbol{v}_2\boldsymbol{\mu}}$.

4.4 Summary

This chapter has described a number of different methods of coding features, both specific and non-specific. The notion of a *feature vector* has been introduced. Feature vectors are used heavily throughout the rest of this thesis.

The choice of which feature coding method to use is entirely dependent upon the task in hand. The following factors should be considered when making a decision:

- What is the type of feature to be coded, is it specific (i.e. a line or a corner) or non-specific?
- Is it required that the feature vectors are invariant to lighting or rotation?
- What is the amount of computational time available to calculate the feature vectors?

Section 4.3 described three measures of feature similarity. The Mahalanobis distance is the best metric if information about the distribution from which the feature vectors were drawn is available. In the absence of this information the Euclidean distance is a suitable alternative.

Chapter 5

Image Feature Saliency

5.1 Introduction

We are working towards a feature-based method capable of automatically providing a dense correspondence across a set of images. We require the correspondence in order to build statistical models of the object, which can in turn be used to interpret unseen images of similar objects. We will demonstrate our algorithms on images of the human face. Such images can be highly variable and contain many features. In order to calculate a correspondence between a pair of images, we could attempt to locate every feature from the first image in the second, but this would be very time consuming because of the number of potential features. Alternatively however, we could locate a small number of features from the first image in the second, giving a partial correspondence. A dense correspondence could then be estimated using the partial correspondence together with some form of interpolation scheme. This gives rise to one important question; which subset of features should we use to find the partial correspondence?

The accuracy to which we can locate a feature from one image in a second varies greatly depending on its appearance. For example, Figure 5.1 shows a number of features from a human face, some of which are easily identifiable, others are less so. Feature (a) is clearly the left side of the left-hand eye. Feature (b), on the other hand could be from the forehead, the cheeks or the chin. We should choose the subset of features which have the greatest probability of accurately locating their true matches. In the case of faces, points around the eyes mouth and nose are likely to be suitable. These features are those least likely to be confused with other structure in the image.

We define *Salient features* to be those with the lowest probability of being misclassified.

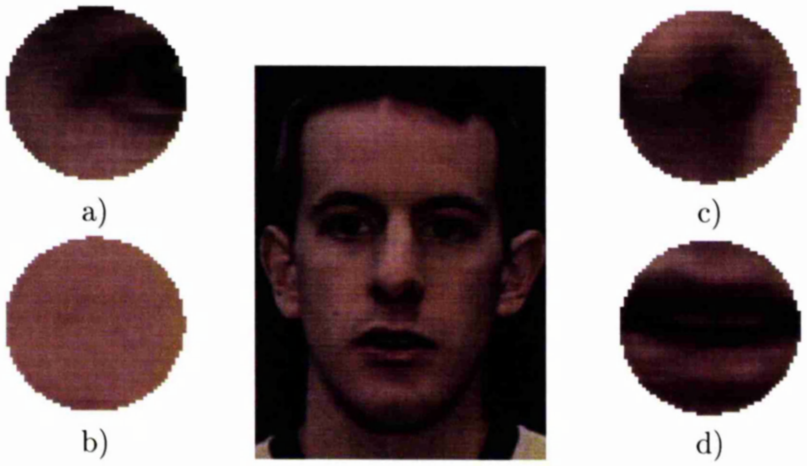


Figure 5.1: Several examples of features extracted from the central face. Note that some of the features (a) are much easier to identify than others (b).

Many authors have faced the problem of which subset of features to use for the purpose of locating some form of correspondence. Lades *et al* [55] used vectors containing the responses of Gabor wavelets at different wavelengths to represent image features. An object was then represented by a set of these feature vectors extracted at the vertices from a sparse grid placed over the object of interest. The vertices of the grid are unlikely to fall over the most salient features, so when searching for the grid in a similar image only the vertices that happened to lie near salient features are found accurately. Triesch *et al* [81] tackles this problem by extracting the feature vectors from the vertices of a graph which is placed manually over the object. The vertices are chosen to coincide with heavily textured positions (i.e. salient features). Although this is an improvement over simply randomly selecting the features, guessing which features will be accurately located is still far from optimal.

In this chapter we explore ways in which the spatial position of salient features can be located automatically in a principled manner.

The aim is to locate salient features, those which are most likely to be found correctly in a subsequent image. Given only one example of the object, the best we can do is to attempt to find those features which are significantly different to all other features in the image containing the object. Ideally, these features would occur exactly once in each example of the object. We define *Image Feature Saliency* to be a measure of how distinctive a feature is compared with the other features in the same image.

For every pixel in the image we construct a feature vector, which describes the structure of the image centered on the pixel at a particular scale σ. The full set of vectors describing features at every pixel forms a multi-variate distribution in a *feature space* (see Figure 5.2). By modelling the density in this feature space we can estimate how likely a given feature is to be confused with other features. We observe that salient features lie in low-density areas of feature space.

The density estimate of feature space at which each feature vector lies corresponds directly to the feature's saliency. The lower the density, the more salient the feature is, since there are few similar features with which it might be confused. A *saliency map* allows us to visualise the saliency measures of all the features in the spatial domain of the original image. An example of a saliency map is shown in Figure 5.2. The peaks of the saliency map correspond with the positions of the most salient features.

We will now describe how to compute image salient features in more detail. In the following sections we review previous literature which has benefited from the notion of

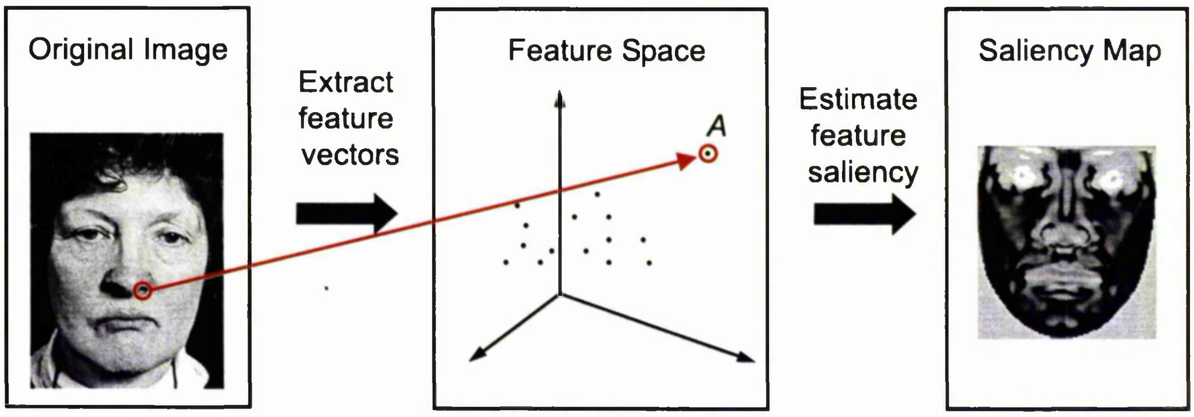


Figure 5.2: An overview of Image Feature Saliency. Feature vector A indicates a salient feature as vector A has a low probability of being confused with other feature vectors.

saliency, describe a number of density estimation methods, how to construct saliency maps and select a subset of salient features from them. Finally results are presented which compare automatically selected salient features with features selected manually.

5.2 Saliency: A background

Many authors have shown that using the saliency of image features can improve the robustness in object recognition algorithms. In this section we will review a number of methods which benefit from the notion of saliency.

5.2.1 Locating salient segments of an object's boundary

Turney *et al* [83, 64] tackled the problem of recognising an object from a partially occluded boundary image. Figure 5.3 shows an example problem. Turney uses a database of object geometry to extract boundary templates from every stable position of every expected object. A set of sub-templates which most differentiate the objects is found. These sub-templates represent the object's *salient* features. Each of the sub-templates is also given a saliency measure which indicates the degree to which the feature is salient.

Given a new scene containing a number of partially occluded object boundaries, matches are located between the sub-templates from the database and the object boundaries segments in the scene. Each match is given a *matching coefficient* which is a measure of the match between the sub-template and a boundary image segment. Merlin and Farber's [63] generalised Hough transform is used to locate plausible centroids of an object. Each sub-template match increments an accumulator in Hough space by its matching coefficient weighted by the sub-template's saliency measure (i.e. matches with salient sub-templates are weighted more heavily). Object matches are local maxima in Hough space. This means that objects which are almost entirely occluded can be located so long as at least one distinguishing (salient) feature is visible. This is a good example of how incorporating the saliency of object features in recognition tasks can increase robustness.

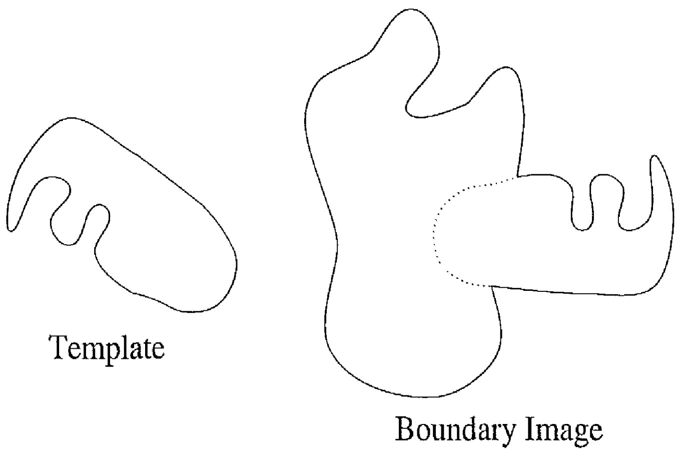


Figure 5.3: Example of a boundary template and a boundary image containing one partially occluded example of the template.

5.2.2 The Local Feature Focus Method

Bolles *et al* [6] propose a method of locating partially visible two-dimensional objects. They locate interesting features such as holes and corners from CAD models of the objects of interest. These features are known as *focus features*. The system designer decides which features would make good focus features. A model is made for each object which consists of the relative geometry of the object's focus features. For each focus feature a strategy is devised which lists its nearby features. Given a new scene all occurrences of the focus features are found and passed on to the hypothesis generation stage. The hypothesis generation stage locates clusters of focus features which correspond to an object model. They generate graphs whose nodes represent possible assignments between object features and image features. Two nodes are connected if their assignments are mutually consistent. A hypothesis for an object's position is found by locating the largest set of mutually consistent clusters of nodes, or the largest fully connected subgraph (maximum cliques). By selecting more than one hypothesis they show it is possible to locate a number of occluded objects in poor quality images. Like Turney (Section 5.2.1), Bolles exploited the fact that it is not necessary to match every feature within an object in order to identify it, a subset of the most salient features in a legal configuration will often suffice. One of the main problems with this approach is that the training phase is not automatic and relies on considerable human interaction, for example in deciding which features are salient and should become focus features.

5.2.3 3DPO: A System for Matching 3-D Objects in Range Data

Bolles and Horaud [7] extended Bolles and Cain's Local Feature Focus method to locate instances of 3D objects in a cluttered scene. Instead of using the standard grey-scale intensity image they use a height image captured from a range sensor.

The focus features used for 3DPO were circular and straight edges. They attempted to manually select focus features by evaluating the following properties:

- Uniqueness
- Cost of detection
- Expected contribution to the identification of the object identity and its position
- Likelihood of detection (is the feature often obscured?)

They note that the selection of further features for use in strategies is even more complex, as not only is it a function of the feature's own characteristics, but also the characteristics of previously located features.

Although this work identified the importance of selecting the correct focus features (salient features), they also realised how complex it is to do manually, even for relatively simple ridged objects. They recognised the need for automated methods of feature selection, which is the problem we address in this chapter.

5.2.4 Locating distinctive groups of symmetry chords

Bailes *et al* [3] proposed a scheme for locating distinctive grey scale regions which lie inside an object boundary. The grey scale regions are represented by samples which lie on chords beginning and ending on the objects boundary as shown in Figure 5.4.

To limit the number of chords which exist only symmetry chords are used. In Figure 5.5 Φ_{ab} is the angle between the boundary normal and the chord AB, the chord AB is only a symmetry chord if $\Phi_{ab} = \Psi_{ab}$.

A vector, $\boldsymbol{g}$, is extracted from each chord which represents the grey levels along it. The grey levels are sampled at equally spaced intervals along the chord.

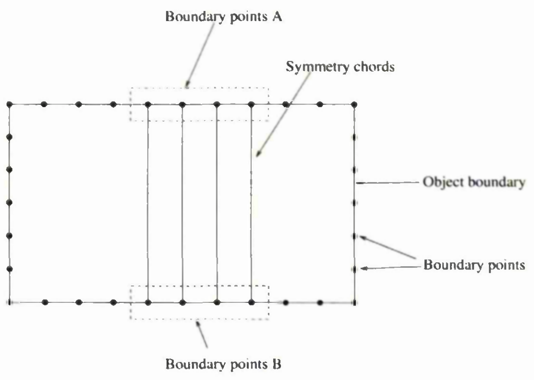


Figure 5.4: A rectangular object and the group of symmetry chords used to represent the grey scales between the sets of boundary points A and B.

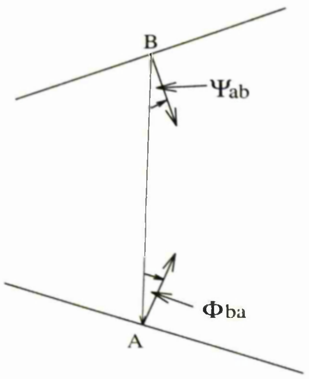


Figure 5.5: The definition of a symmetry chord AB.

Neighboring chords are then merged into groups if there is little variation in their grey level vectors. The mean $\bar{\boldsymbol{g}}$, and covariance matrix $\boldsymbol{S}$ can then be calculated and used to determine a measure of distinctiveness for each group.

Bailes then goes on to describe how the distinct groups of symmetry chords can be used to achieve robust object recognition.

5.3 Density Estimation

Our definition of Image Feature Saliency relies on accurately estimating the probability density function from a set of samples (in our case, the features from all pixels in an image). Here we review the different approaches.

Density estimation is the construction of a probability density function from some observed data, in this case feature vectors. The different density estimation approaches fall into one of two categories. *Parametric* schemes assume that the data is drawn from a known parametric family of distributions. The data is used to estimate optimal values of the parameters of the distribution. For example, if it was assumed that the data comes from a normal distribution estimating the density of would involve finding estimates for the mean and covariance of the distribution. The *nonparametric* approach makes less rigid assumptions about the distribution of observed data. Silverman [13] describes a number of non-parametric density estimation techniques based around placing kernels at each data point in detail. We consider a number of both parametric and non-parametric density estimation techniques in the following sections.

5.3.1 Parametric Methods

Gaussian models

A frequently used approach is to approximate the density of feature space with a single multivariate Gaussian. An estimate of the local density $\hat{p}$ at point $\boldsymbol{x}$ in feature space is given by:

$$\hat{p}(\boldsymbol{x}) = G(\boldsymbol{x}|\boldsymbol{\mu};\boldsymbol{S}) = (2\pi)^{-\frac{N}{2}}|\boldsymbol{S}|^{-\frac{1}{2}}e^{-(\boldsymbol{x}-\boldsymbol{\mu})^T\boldsymbol{S}^{-1}(\boldsymbol{x}-\boldsymbol{\mu})} \tag{5.1}$$

where N is the dimensionality of the distribution, $\boldsymbol{S}$ is the covariance matrix of the entire distribution and $\boldsymbol{\mu}$ is the distribution mean.

The advantage of this method is that the complexity of calculating the density estimates for all feature vectors is of order n, where n is the number of feature vectors in feature space. Figure 5.6(a) shows an example of data and a probability density function for which this approach is suitable.

This method's main weakness becomes apparent when considering the result of applying it to a distribution with a multimodal probability density function. Figure 5.6(b) shows a situation where the feature space is formed by two subdistributions, the mean of which lies part way between them. Approximating this with a single Gaussian results in the distribution mean being the region of greatest density, which is clearly a bad approximation.

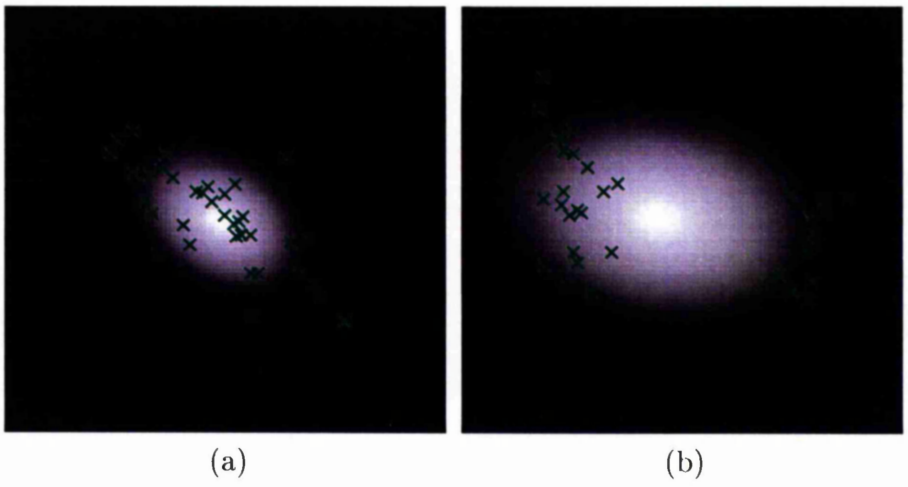


Figure 5.6: Examples of density estimation using a single Gaussian. The crosses gives the data, the image gives the estimated gaussian probabilty density function. (b) shows an example where modelling the density with a single Gaussian gives inadequate estimation.

Mixture models

Mixture models attempt to estimate the density of a distribution with a mixture of kernels, often Gaussian in form. The parameters for each kernel are first initialised using a simple scheme such as distributing the kernels evenly around the space. The parameters can then be optimised to fit the data using the Expectation Maximisation algorithm.

Expectation Maximisation (EM) is an iterative optimisation technique, originally proposed by Dempster, Laird and Rubin [28]. EM can be used as a method to optimise the parameters of a mixture of multi-variate Gaussians to best fit a data set [62]. The method proceeds from a starting set of parameters using two steps.

- **Expectation:** Calculate the probability p_{ki} that the sample $\boldsymbol{x}_i$ belongs to the Gaussian mixture model component k. p_{ki} is given by:

$$p_{ki} = \frac{w_k.G(\boldsymbol{x}_i|\boldsymbol{\mu}_k;\Sigma_k)}{\sum_j w_j.G(\boldsymbol{x}_i|\boldsymbol{\mu}_j;\Sigma_j)} \tag{5.2}$$

- **Maximisation:** calculating new estimates of the parameters μ_k (mean vector), Σ_k (covariance matrix) and w_k (mixing fraction). μ_k, Σ_k and w_k are given by:

$$w_k = \frac{1}{n}.\sum_{i=1}^{n} p_{ki} \tag{5.3}$$

$$\mu_k = \frac{1}{n}.\sum_{i=1}^{n} \frac{p_{ki}\boldsymbol{x}_i}{w_k} \tag{5.4}$$

$$\Sigma_k = \frac{1}{n}.\sum_{i=1}^{n} \frac{p_{ki}(\boldsymbol{x}_i - \mu_k)(\boldsymbol{x}_i - \mu_k)^T}{w_k} \tag{5.5}$$

These two steps are repeated until the method converges to an optimal set of parameters. The local probability density estimation $\hat{p}$ at $\boldsymbol{x}$ is then given by:

$$\hat{p}(\boldsymbol{x}) = \sum_{k=1}^{n_s} w_k G(\boldsymbol{x}|\boldsymbol{\mu}_k;\Sigma_k) \tag{5.6}$$

This approach suffers because it is necessary to select the number of kernels with which to model the data's density. It can also be difficult to choose the initial starting positions and parameters of the kernels. Poor initialisation can result in a bad final approximation. For low dimensional data, initialisation schemes resulting in good solutions can be employed, but for higher dimensional data this becomes a more challenging problem.

5.3.2 Nonparametric Methods

Kernel Method

The Kernel method positions kernels (e.g. Gaussians) at all the samples in the distribution. The local density $\hat{p}$ at point $\boldsymbol{x}$ in a distribution is estimated by summing the contribution from a mixture of kernels, in this case gaussians:

$$\hat{p}(\boldsymbol{x}) = \frac{1}{n}\sum_{i=1}^{n} G(\boldsymbol{x}|\boldsymbol{x}_i; h\boldsymbol{S}) \tag{5.7}$$

where n is the number of samples in the distribution, $\boldsymbol{x}_i$ is the i^{th} sample from the distribution, $\boldsymbol{S}$ is the covariance of the whole distribution and h is a scaling factor. Silverman [13] suggests a value for h given by:

$$h = \left[\frac{4}{N(D+2)}\right]^{\frac{1}{(D+4)}} \tag{5.8}$$

where D is the number of dimensions. This method gives an accurate estimation of the distribution's density but is very time consuming to calculate for large n as the algorithm's complexity is of order n.

Sub-sampled Kernel method

This method attempts to approximate the Kernel method by placing gaussian kernels at a randomly selected n_s of the original n points. (5.8) suggests h increases as n_s decreases. Evaluation of the probability density function is order n_s (where $n_s < n$), so can be much more efficiently calculated than the full kernel method, but this comes with some loss of accuracy.

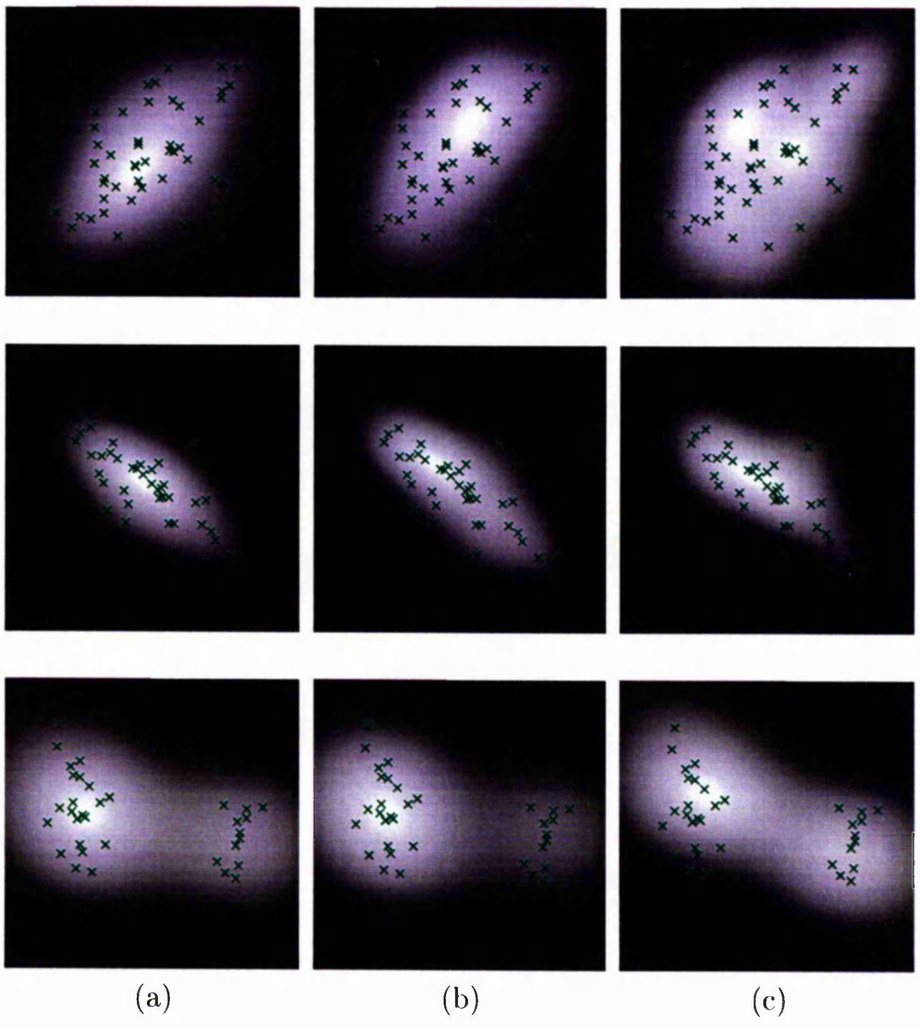


Figure 5.7: Comparison of the density estimate obtained from several distributions, using the kernel method (a) and the random kernel method with 50% (b) and 25% (c) of the kernels. Each row shows a different set of raw data.

5.3.3 Comparison of Density Estimation methods

Approximating with a single Gaussian is fast but is not valid in situations where the data is not approximately normal.

The Kernel method gives the best approximation but at the cost of being computationally intensive. The Expectation maximisation algorithm and the Sub-sampled Kernel method both try to approximate the Kernel method by using fewer Gaussians than samples. Expectation maximisation takes a long time in comparison to the Sub-sampled Kernel method to achieve results similar to that of the Kernel method. A single iteration of the expectation step has the same complexity as the sub-sampled kernel method, i.e. n_s. However, it is likely to give a better estimate of the density. The Expectation maximisation algorithm also has a number of issues regarding the initialisation of the kernels. For these reasons the Sub-sampled Kernel approach is used as the method of choice.

In Figure 5.7 we compare the density estimate obtained from several distributions, using the kernel method (a) and the Sub-sampled kernel method with 50% (b) and 25% (c) of the kernels (randomly selected).

5.4 Saliency Measures and Saliency Maps

The density estimate for each feature vector $\boldsymbol{v}_{\boldsymbol{x}\sigma}$ leads directly to the saliency of the feature at scale σ. The lower the density, the more salient the point. The saliency measure, $s(\boldsymbol{x}|\sigma)$, of feature vector $\boldsymbol{v}_{\boldsymbol{x}\sigma}$ is given by:

$$s(\boldsymbol{x}|\sigma) = -\hat{p}(\boldsymbol{v}_{\boldsymbol{x}\sigma}) \tag{5.9}$$

Thus a low $\hat{p}$ leads to a high saliency measure $s_{\boldsymbol{x}\sigma}$.

A saliency map is a way of visualising the saliency measures for a particular scale. It is constructed by setting each pixel from the original image to the saliency of the feature vector centered at that point.

We can efficiently obtain saliency maps for a number of scales by constructing a Gaussian image pyramid [12] and calculating a saliency map for the image at each level in the pyramid. Figure 5.8 illustrates saliency maps extracted from the first 3 levels of a Gaussian image pyramid. The feature vectors contained the first, second and third order Cartesian differential invariants (see Section 4.2.2), making them 8 dimensional.

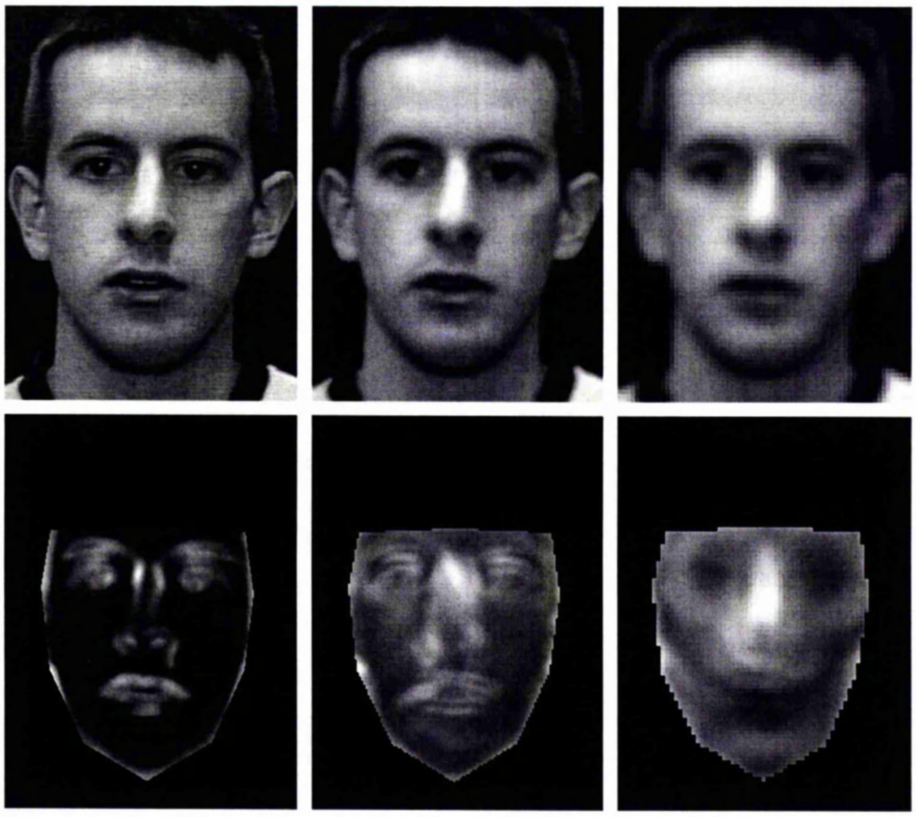


Figure 5.8: The saliency maps extracted from the first 3 levels of a Gaussian image pyramid (fine to coarse scale going from left to right). Bright regions are the most salient.

5.5 Selecting a subset of salient features

The spatial positions of the most salient features can be found by locating the peaks in each the saliency map. A peak is defined in this thesis as pixel which is higher than its 8 immediate neighbors. Selecting the n highest peaks often results in all salient features being clustered around a small area. To ensure that the features selected are distributed evenly across the object, the following algorithm is used:

```
REPEAT
    Select the next highest peak in the saliency map
    IF the peak is more than $\frac{p}{2L}$ pixel's from any previously selected trough
        THEN Mark peak as a representing a salient feature
    ELSE
        Discard peak
    ENDIF
UNTIL no more peak's
```

where L is the level of the Gaussian image pyramid. $L = 0$ is the original image and $L = 1$ is an image of half the size (quarter the area). p is the minimum separation in pixels of the salient features in the original image ($L = 0$). Figure 5.9 shows the salient features selected at the first 3 levels of a Gaussian image pyramid. The top row of images shows the highest 25 peaks. Note how clustered the selected features are. The bottom row shows the result of the application of the above algorithm, the chosen points are distributed much better.

5.6 Results

In order to quantify the usefulness of the selected salient features, we addressed the question "how successfully can we find the salient features in unseen images?".

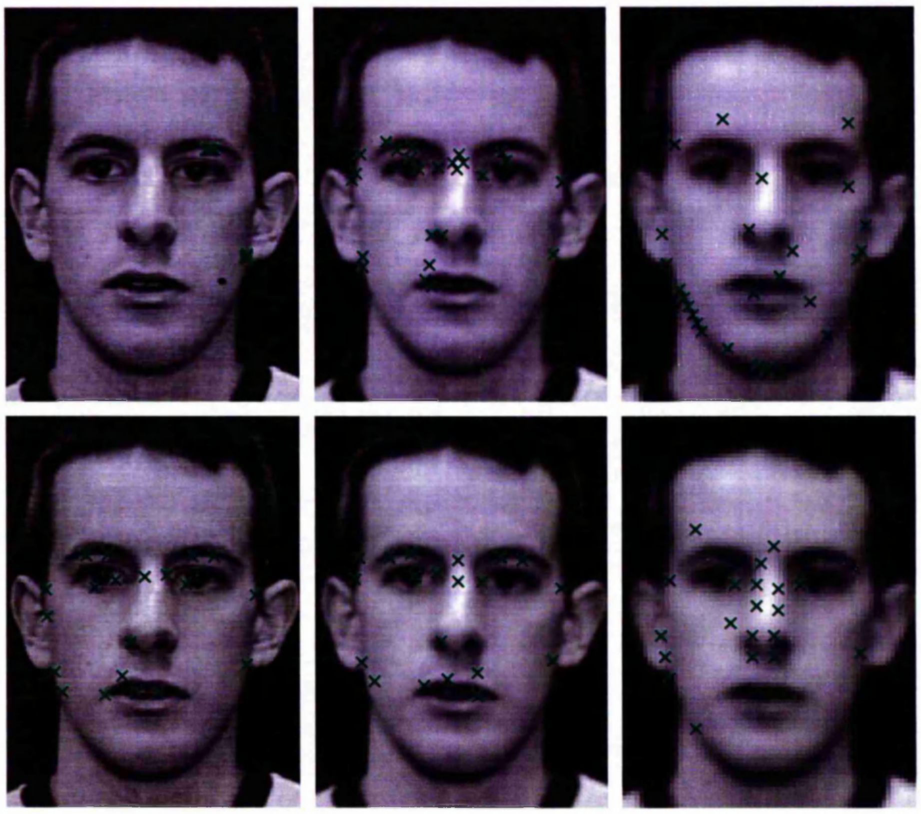


Figure 5.9: The salient features selected from the first 3 levels of a Gaussian image pyramid (fine to coarse scale going from left to right). The top row shows the 25 highest peaks, the bottom row shows the effect of applying the algorithm highlighted in Section 5.5.

We located the 20 most salient features (Figure 5.11(a)) and attempted to locate these features in 188 unseen images of human faces. 1^{st}, 2^{nd} and 3^{rd} order Cartesian Differential Invariants [38] were used in the feature extractor. Figure 5.10* shows a number of examples of the 188 images used. For each search the number of false positive matches was recorded. For example, if the true match is the third most likely match then two false positive matches would be recorded.

To provide a means of contrast we also selected 20 features by hand (Figure 5.11(b)) and 20 randomly selected features (Figure 5.11(c)), and repeated the test on these. The features that were selected by hand correspond with those which are typically

*The images used in this experiment were kindly provided by Dr Bob Nicholls, PITO, UK.

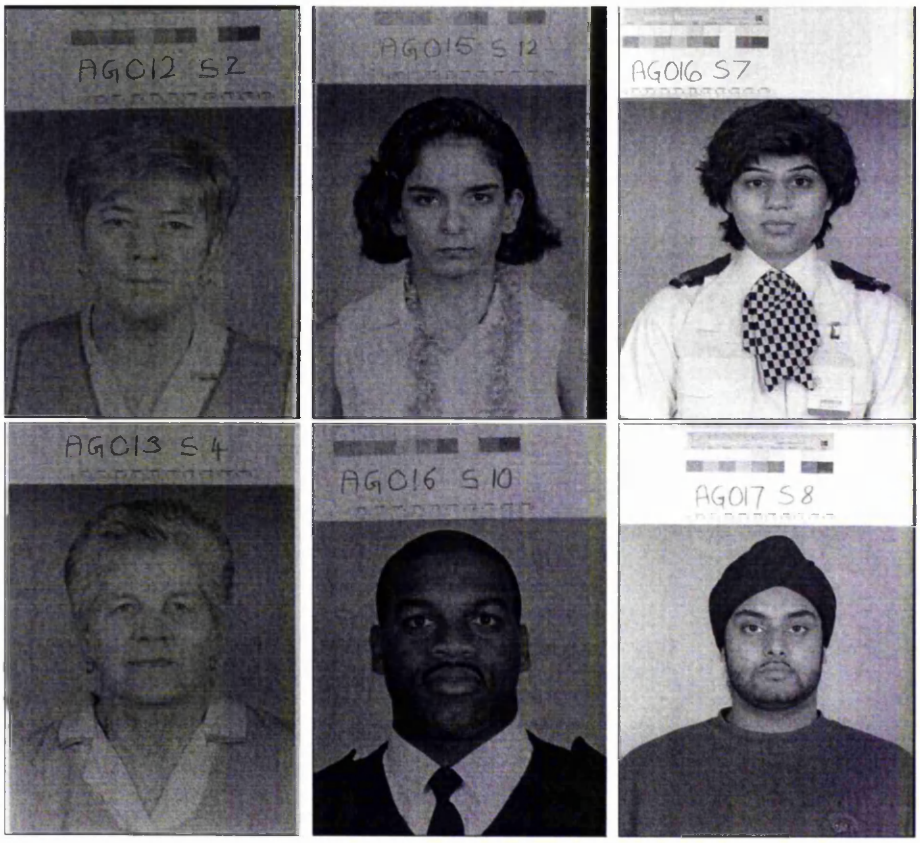


Figure 5.10: Examples of the images used to obtain the results presented in Section 5.6

selected for the purpose of building statistical models of the human face.

The results of the tests are shown in Figure 5.12. The graph shows the percentage of successful searches according to the number of acceptable false positives. A search is said to be successful if the true match is found within, say, the 10 most probable matches (i.e. in this case the number of acceptable false positives is 10). It can be seen that, as expected, the randomly selected features are located least successfully. Locating hand selected features approximately doubles the performance compared to randomly selected features. Most importantly, the graph also shows that salient features are more likely to be accurately found than either randomly selected or hand selected features. The percentages shown in the graph are based on 11280 searches.

The time taken to calculate a saliency image of the mean face (approximately

8500 pixels) using the Sub-sampled Kernel method with 100 kernels is approximately 25 seconds on a Sun Ultra 2. The time is directly proportional to the number of kernels, so if 1000 kernels are used this increases to 250 seconds.

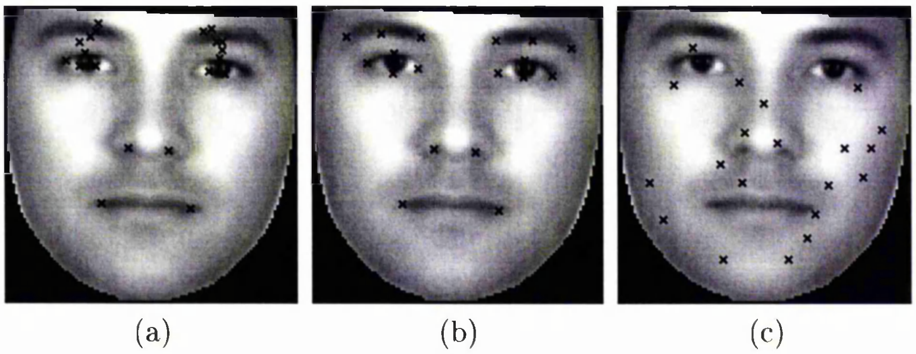


Figure 5.11: The most salient features (a), hand selected features (b) and randomly selected features (c). These are the features used in the results.

5.7 Discussion

We have described how a probabilistic measure of saliency can be used to select those object features from a single example of the object which are least likely to generate false positive matches when searching a new image of the object. The salient features are selected by evaluating the *Image Feature Saliency* of every pixel in a single image. They are chosen semi-independently and are therefore not the optimum group. Finding the optimum group is an extremely complicated task, as stated by Bolles *et al* [6].

We have also presented quantitative results showing that salient features can be found with a greater degree of success in unseen object examples than randomly selected features or manually selected features.

We have applied the notion of saliency to improve the robustness of one object

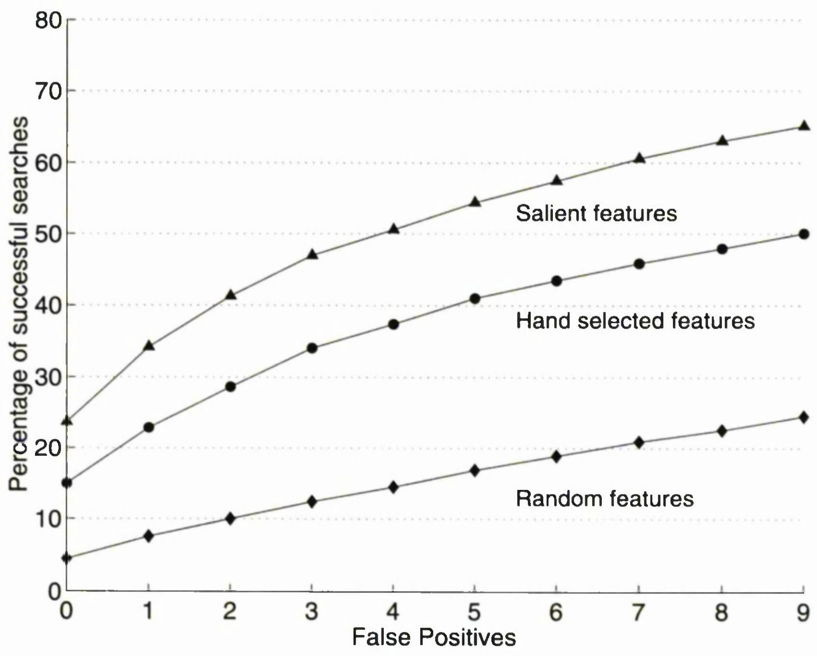


Figure 5.12: Graph illustrating the percentage of successful searches according to the number of false positives. The graph can be interpreted in the same manner as a ROC curve, i.e. the closer the curve is to the top left, the better the result.

interpretation task, locating new instances of a human face. We also believe that other interpretation tasks can also benefit from the application of saliency.

The ideas presented in this chapter have been included in a number of conference and journal publications [85, 89, 88].

Chapter 6

Object Feature Saliency

6.1 Introduction

In the previous chapter we showed that locating salient features using one training example results in features which can be found in unseen object examples more reliably than hand chosen features. When training on only a single example, we define a salient feature to be one which is significantly different to all others in that image. We call this Image Feature Saliency. However, this does not take account of how *reliable* the feature is, whether it occurs in all examples of the object or whether it varies in a way which could cause it to be confused with other object features. These factors can not be determined by considering just a single example. In this chapter we describe selecting salient features by analysing several image examples. We call a measure derived from a set of images *Object Feature Saliency.* Object Feature Saliency is a property of the object class rather than of a particular image.

As an illustration, consider Figure 6.1. Figure 6.1(a) shows a face image where the nostrils appear as black circular features. Saliency analysis on this image alone would choose the nostrils as salient features. Figure 6.1(b) shows a second face image with a slightly different pose. The head is tilted slightly forward, drastically changing the appearance of the nostrils. In this second example the appearance of the nostrils could easily be confused with the appearance of many other features within the face. By considering both of these images it is possible learn more about how the nostril varies. This information is useful for deciding if the nostril is a reliable salient feature.

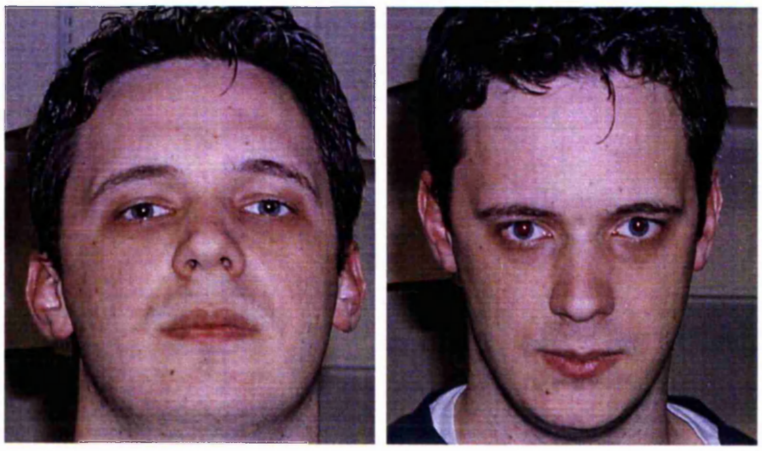


Figure 6.1: An example of how features can vary. (a) shows the nostrils as dark circle regions, in (b) a slight change in head pose drastically changes the nostrils appearance.

In this chapter we present a new approach for locating salient features, which attempts to model how each individual feature varies over a number of training examples for which a correspondence exists. We extract a feature vector describing a particular feature from all training examples. The set of feature vectors which describe the feature is then used to create a feature model. The saliency of the feature is determined by using the models for many features to calculate the probability of mis-classifying any one feature as any other feature.

In the following sections we define how the feature models are built and how the saliency measures are calculated from the feature models.

6.2 Building Feature Models

In order to extract a feature vector representing the same feature in all training examples, we first establish a correspondence between all training examples. To this end, we place a set of consistent landmarks on each of the training examples and use them as control points for an interpolation scheme. Figure 6.2 shows some examples of such landmarks placed on faces. In our experiments these points have been placed by hand.

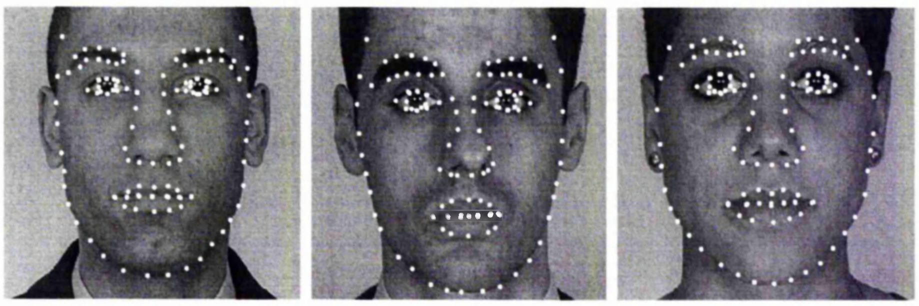


Figure 6.2: Examples of face images with common landmarks.

More formally, let $\boldsymbol{x}_j = f_{ji}(\boldsymbol{x}_i|\boldsymbol{y}_i, \boldsymbol{y}_j)$ be a mapping from points $\boldsymbol{x}_i$ in image i to points $\boldsymbol{x}_j$ in image j, controlled by a list of control points $\boldsymbol{y}_i$ and $\boldsymbol{y}_j$. We adopted a piece-wise affine interpolator to define this mapping, although a thin-plate spline interpolator [8] would give a smoother answer at the expense of additional computational complexity. For further details on interpolation see Appendix A.

In order to define the features we wish to model, we calculate a mean face [32] based on the training examples. We then model one feature for each pixel in the mean face image. The approximate number of pixels in the mean face, n_p, can be set according to the computational power available by varying its scale.

Let $\boldsymbol{v}_{ik}$ be the feature vector extracted around the k^{th} pixel in the ith image. The spatial position of the center from which the k^{th} feature vector is extracted in image

i is given by:

$$\boldsymbol{x}_{ik} = f_{i,mean}(\bar{\boldsymbol{x}}_k | \boldsymbol{l}_{mean}, \boldsymbol{l}_i) \tag{6.1}$$

where $\bar{\boldsymbol{x}}_k$ gives the position of the k^{th} pixel in the mean face, $\boldsymbol{l}_i$ is the positions of the consistent landmarks from image i and $\boldsymbol{l}_{mean}$ gives the positions of the consistent landmarks in the mean image.

The probability density function for feature k, $p_k(\boldsymbol{v})$, is then modelled using a multi-variate gaussian with mean $\boldsymbol{\mu}_k$ and covariance $\boldsymbol{\Sigma}_k$, given by:

$$\boldsymbol{\mu}_k = \frac{\sum_{n_t}^{i=1} \boldsymbol{v}_{ik}}{n_t} \tag{6.2}$$

$$\boldsymbol{\Sigma}_k = \frac{\sum_{n_t}^{i=1} (\boldsymbol{v}_{ik} - \boldsymbol{\mu}_k)^T (\boldsymbol{v}_{ik} - \boldsymbol{\mu}_k)}{n_t - 1} \tag{6.3}$$

where n_t is the number of training examples.

The parameters $\boldsymbol{\mu}_k$ and $\boldsymbol{\Sigma}_k$ together define the feature model for feature k.

6.3 Calculating a Feature's Saliency

Given a probability density function, $p_k(\boldsymbol{v})$, for each feature (in this case a Gaussian with mean $\boldsymbol{\mu}_k$ and covariance $\boldsymbol{\Sigma}_k$) it is now possible to calculate a feature's saliency. The saliency of feature k, s_k, is given by the probability of not mis-classifying feature

k with any other object feature. Thus:

$$s_k = 1 - \frac{1}{n_p - 1} \sum_{j=1;j\neq k}^{n_p} \epsilon(p_k(\boldsymbol{v}), p_j(\boldsymbol{v})) \tag{6.4}$$

where $\epsilon(p_k(\boldsymbol{v}), p_j(\boldsymbol{v}))$ is the probability of misclassifying feature j with feature k. So in order to evaluate the saliency of an object's features it is necessary to evaluate $\epsilon(p_k(\boldsymbol{v}), p_j(\boldsymbol{v}))$. $p_k(\boldsymbol{v})$ and $p_j(\boldsymbol{v})$ will generally represent multidimensional distributions, but in order to understand the problems which arise when calculating $\epsilon(p_k(\boldsymbol{v}), p_j(\boldsymbol{v}))$, we will first explain how to calculate $\epsilon(p_k(\boldsymbol{v}), p_j(\boldsymbol{v}))$ in the 1D case. It will become clear that the analytic 1D solution is difficult to extend to multiple dimensions.

Calculating $\epsilon(p_k(\boldsymbol{x}), p_j(\boldsymbol{x}))$ in the 1D case

Consider first the 1D case. We wish to calculate $\epsilon(p_k(\boldsymbol{x}), p_j(\boldsymbol{x}))$, the probability of misclassifying a sample from distribution $p_k(\boldsymbol{x})$ as being from $p_j(\boldsymbol{x})$. In the case where $p_k(\boldsymbol{x})$ and $p_j(\boldsymbol{x})$ are Gaussians, an analytic solution using error functions exists.

In the one dimensional space, the probability of a sample x being generated from the probability density function for feature k is given by:

$$p_k(x) = \frac{1}{\sqrt{2\pi{\sigma_k}^2}} e^{-\frac{(x-\mu_k)^2}{2{\sigma_k}^2}} \tag{6.5}$$

where σ_k and μ_k are the standard deviation and mean determined from the training set.

Assuming all features are equally likely, the Likelihood ratio test suggests classi-

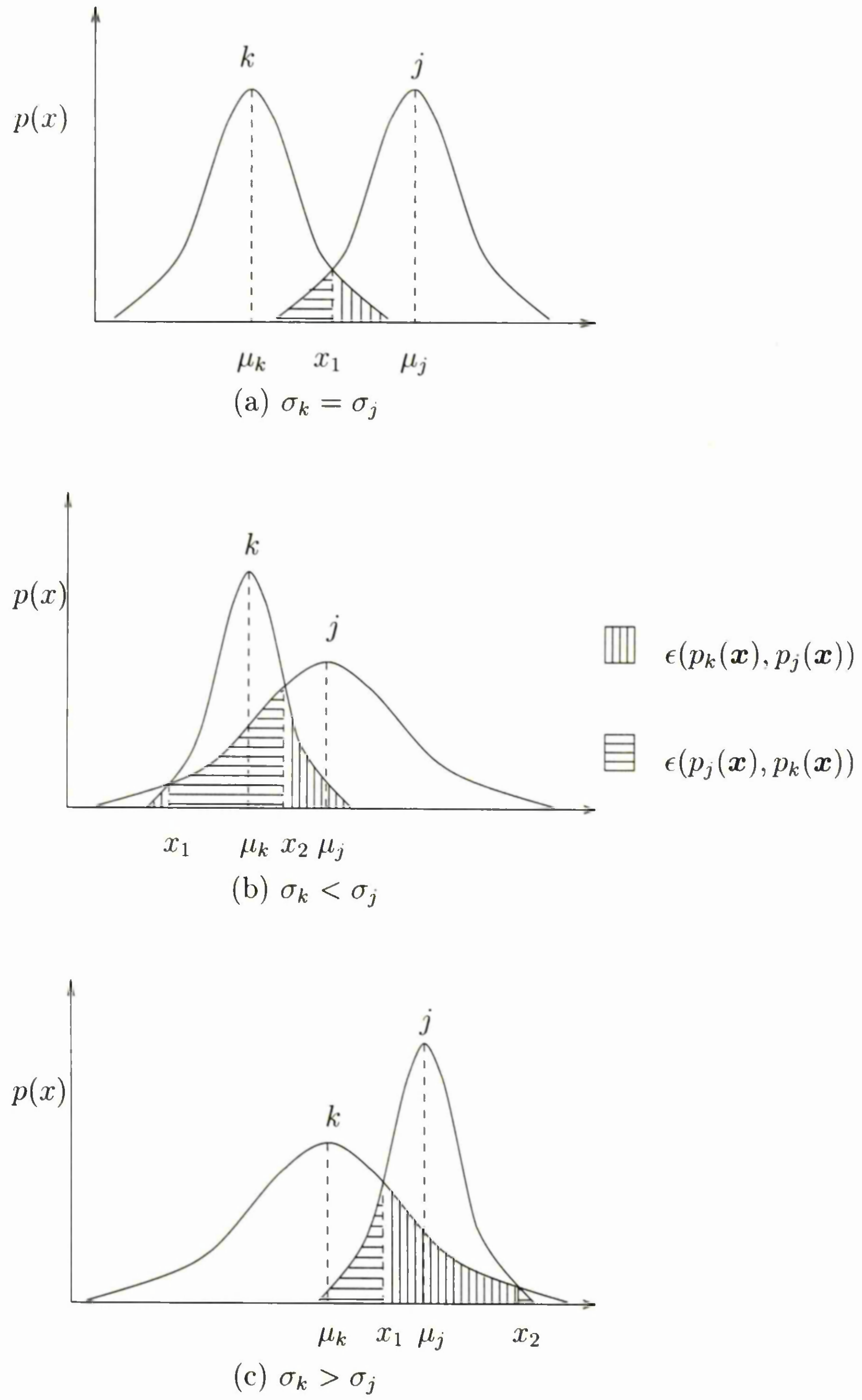

$p(x)$
k
j
μ_k x_1 μ_j
(a) $\sigma_k = \sigma_j$
$p(x)$
k
j
x_1 μ_k x_2 μ_j
(b) $\sigma_k < \sigma_j$
▥ $\epsilon(p_k(\boldsymbol{x}), p_j(\boldsymbol{x}))$
▤ $\epsilon(p_j(\boldsymbol{x}), p_k(\boldsymbol{x}))$
$p(x)$
j
k
μ_k x_1 μ_j x_2
(c) $\sigma_k > \sigma_j$

Figure 6.3: Illustrates the areas which need to be calculated in order to evaluate $\epsilon(p_k(\boldsymbol{x}), p_j(\boldsymbol{x}))$ in the 1D case. There are three cases depending on the values of σ_k and σ_j. Without loss of generality, assume $\mu_k < \mu_j$.

fying x as class k if $p_k(x) > p_j(x)$. This splits the space up into two or three regions, where the boundary points, x_b, satisfy:

$$p_k(x_b) = p_j(x_b) \tag{6.6}$$

Figure 6.3 illustrates these regions. Without loss of generality we assume $\mu_k < \mu_j$. There are three situations depending on the values of σ_k and σ_j. Taking logs from equation 6.6 results in a quadratic as shown:

$$\begin{aligned}
\frac{e^{-\frac{(x-\mu_k)^2}{2\sigma_k{}^2}}}{e^{-\frac{(x-\mu_j)^2}{2\sigma_j{}^2}}} &= \frac{\sigma_k}{\sigma_j} && (6.7)\\
\frac{(x-\mu_j)^2}{2\sigma_j{}^2} - \frac{(x-\mu_k)^2}{2\sigma_k{}^2} &= ln(\frac{\sigma_k}{\sigma_j}) && (6.8)\\
ax^2 + bx + c &= 0 && (6.9)
\end{aligned}$$

where

$$\begin{aligned}
a &= \frac{1}{\sigma_j{}^2} - \frac{1}{\sigma_k{}^2}\\
b &= 2(\frac{\mu_k}{\sigma_k{}^2} - \frac{\mu_j}{\sigma_j{}^2})\\
c &= \frac{\mu_j{}^2}{\sigma_j{}^2} - \frac{\mu_k{}^2}{\sigma_k{}^2} - 2ln(\frac{\sigma_k^2}{\sigma_j^2})
\end{aligned}$$

Solving the quadratic gives:

$$x_1 = \frac{-b + \sqrt{b^2 - 4ac}}{2a} \qquad x_2 = \frac{-b - \sqrt{b^2 - 4ac}}{2a} \tag{6.10}$$

$\epsilon(p_k(\boldsymbol{x}), p_j(\boldsymbol{x}))$ can now be approximated in one of the three following ways, depending on the value of σ_k and σ_j:

if $\sigma_k = \sigma_j$ then

$$\epsilon(p_k(\boldsymbol{x}), p_j(\boldsymbol{x})) = \frac{1}{2}(1 - erf(\frac{x_1 - \mu_k}{\sqrt{2}\sigma_k})) \tag{6.11}$$

if $\sigma_k < \sigma_j$ then

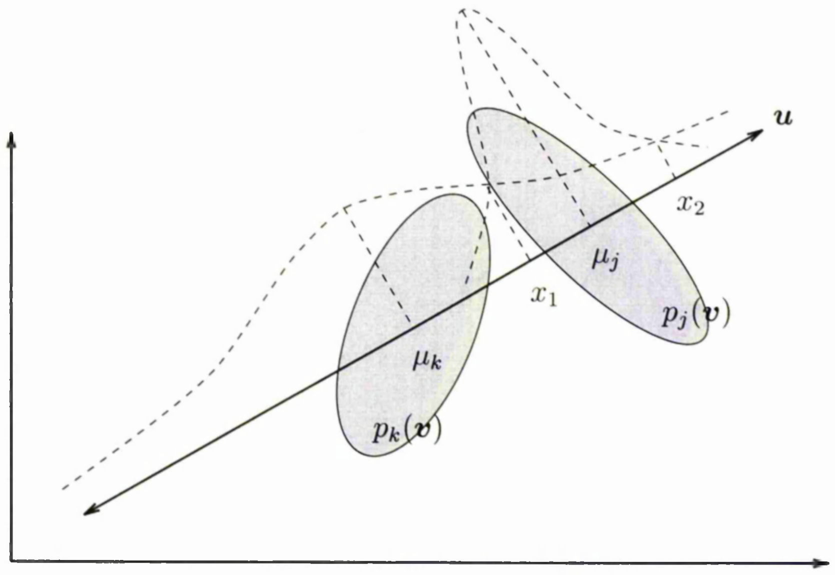
$$\epsilon(p_k(\boldsymbol{x}), p_j(\boldsymbol{x})) = 1 + \frac{1}{2}(erf(\frac{x_1 - \mu_k}{\sqrt{2}\sigma_k}) - erf(\frac{x_2 - \mu_k}{\sqrt{2}\sigma_k})) \tag{6.12}$$

if $\sigma_k > \sigma_j$ then

$$\epsilon(p_k(\boldsymbol{x}), p_j(\boldsymbol{x})) = \frac{1}{2}(erf(\frac{x_2 - \mu_k}{\sqrt{2}\sigma_k}) - erf(\frac{x_1 - \mu_k}{\sqrt{2}\sigma_k})) \tag{6.13}$$

Thus, equations 6.11, 6.12 and 6.13 can then be substituted into equation 6.4 to obtain a measure of feature saliency.

Typically the dimensionality of the feature vectors is much higher than 1. With higher dimensions the misclassification regions become increasingly complicated volumes, making $\epsilon(p_k(\boldsymbol{x}), p_j(\boldsymbol{x}))$ hard to calculate. Also, to calculate the saliency measure for all features, $\epsilon(p_k(\boldsymbol{x}), p_j(\boldsymbol{x}))$ must be evaluated approximately $\frac{1}{2}(n_p)^2$ times. Because of the complexity of calculating $\epsilon(p_k(\boldsymbol{x}), p_j(\boldsymbol{x}))$ in high dimensional spaces and the large number of times it must be evaluated it is necessary to approximate $\epsilon(p_k(\boldsymbol{x}), p_j(\boldsymbol{x}))$ for spaces with more than one dimension.



Feature space, $\boldsymbol{v}$

Figure 6.4: Illustration of how a multi-variate Gaussian classification can be approximated by a single dimensional problem.

Approximating $\epsilon(p_k(\boldsymbol{v}), p_j(\boldsymbol{v}))$ in multi dimensional feature spaces

We approximate $\epsilon(p_k(\boldsymbol{v}), p_j(\boldsymbol{v}))$ by simplifying the problem to a single dimension, where it can be solved using error functions as in Section 6.3. This process is illustrated in Figure 6.4. The first step is to construct a new one dimensional space. This is simply the axis which passes through the mean of both feature models $p_k(\boldsymbol{v})$ and $p_j(\boldsymbol{v})$. This axis is labelled $\boldsymbol{u}$ in Figure 6.4 and is given by:

$$\boldsymbol{u} = \boldsymbol{\mu}_i + \alpha\, \delta\boldsymbol{u} \tag{6.14}$$

$$\delta\boldsymbol{u} = \frac{\boldsymbol{\mu}_j - \boldsymbol{\mu}_i}{|\boldsymbol{\mu}_j - \boldsymbol{\mu}_i|} \tag{6.15}$$

The variance, σ_k, along $\boldsymbol{u}$ due to distribution $p_k(\boldsymbol{v})$ is given by:

$${\sigma_k}^2 = \delta \boldsymbol{u}^T.\Sigma_k.\delta \boldsymbol{u} \tag{6.16}$$

similarly

$${\sigma_j}^2 = \delta \boldsymbol{u}^T.\Sigma_j.\delta \boldsymbol{u} \tag{6.17}$$

where $\boldsymbol{\Sigma}_j$ is the covariance matrix of feature model $p_j(\boldsymbol{v})$.

μ_k, μ_j, σ_k and σ_j now form a one dimensional problem which can be solved using error functions as shown in Section 6.3.

6.3.1 Monte Carlo trial

In order to determine if the 1D approximation for $\epsilon(p_k(\boldsymbol{v}), p_j(\boldsymbol{v}))$ is a reasonable one, a Monte Carlo trail was carried out as a comparison.

Two 6 dimensional feature models were created, $p_a(\boldsymbol{v})$ and $p_b(\boldsymbol{v})$, defined by;

$$\begin{aligned}
\boldsymbol{\mu}_a &= (0,0,0,0,0,0)^T \\
\Sigma_a &= \boldsymbol{I} \\
\boldsymbol{\mu}_b &= (x,0,0,0,0,0)^T \\
\Sigma_b &= \sigma^2 \boldsymbol{I}
\end{aligned}$$

where $\boldsymbol{I}$ is the identity matrix.

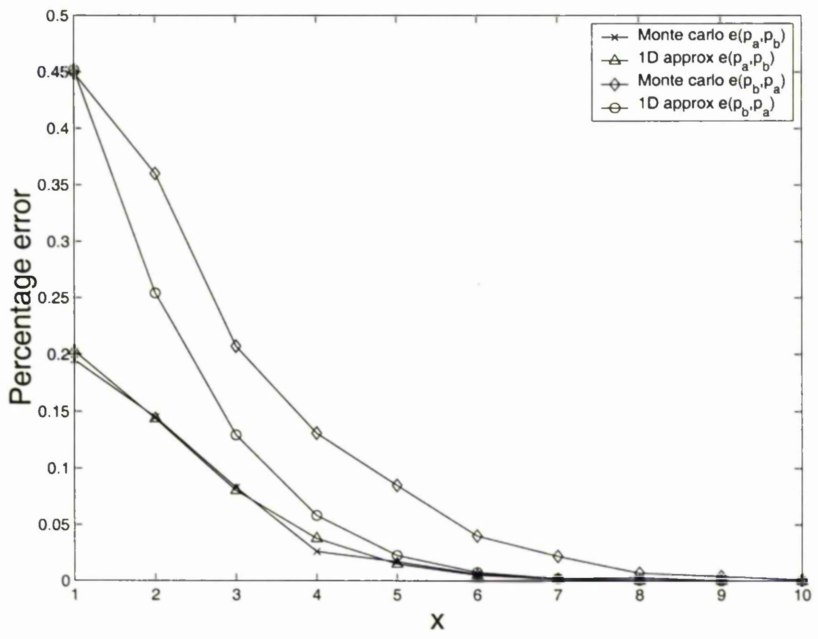


Figure 6.5: Graph showing comparing the error in $\epsilon(p_a(\boldsymbol{v}), p_b(\boldsymbol{v}))$ and $\epsilon(p_b(\boldsymbol{v}), p_a(\boldsymbol{v}))$ for varying values of x ($\sigma = 1$). Results were obtained using a Monte Carlo method and the 1D approximation method.

$\epsilon(p_a(\boldsymbol{v}), p_b(\boldsymbol{v}))$ and $\epsilon(p_b(\boldsymbol{v}), p_a(\boldsymbol{v}))$ where calculated for various values of x and then for various values of σ^2. To estimate the true values, $\epsilon(p_a(\boldsymbol{v}), p_b(\boldsymbol{v}))$ and $\epsilon(p_b(\boldsymbol{v}), p_a(\boldsymbol{v}))$ were approximated using a Monte Carlo trail. This involved generating 10000 random samples from both $p_a(\boldsymbol{v})$ and $p_b(\boldsymbol{v})$. The samples were then classified as belonging to $p_a(\boldsymbol{v})$ or $p_b(\boldsymbol{v})$. A given sample $\boldsymbol{s}_i$ belongs to $p_a(\boldsymbol{v})$ if $c(\boldsymbol{s}_i)$ is equal to 1, otherwise it belongs to $p_b(\boldsymbol{v})$, where:

$$c(\boldsymbol{s}_i) = \begin{cases} 1 & \text{if } (\boldsymbol{s}_i - \boldsymbol{\mu}_a)^T \boldsymbol{\Sigma}_a^{-1} (\boldsymbol{s}_i - \boldsymbol{\mu}_a) - k_a < (\boldsymbol{s}_i - \boldsymbol{\mu}_b)^T \boldsymbol{\Sigma}_b^{-1} (\boldsymbol{s}_i - \boldsymbol{\mu}_b) - k_b \\ 0 & \text{otherwise} \end{cases} \tag{6.18}$$

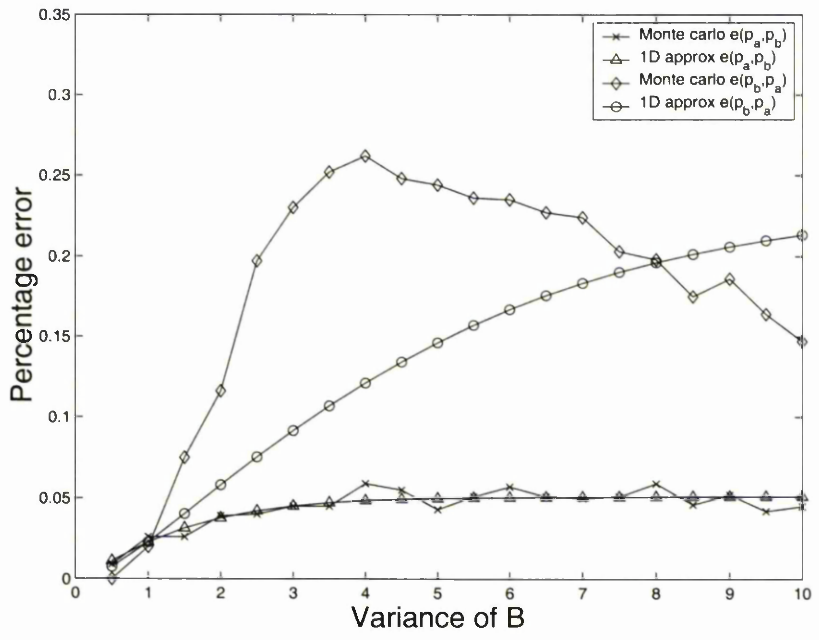


Figure 6.6: Graph showing comparing the error in $\epsilon(p_a(\boldsymbol{v}), p_b(\boldsymbol{v}))$ and $\epsilon(p_b(\boldsymbol{v}), p_a(\boldsymbol{v}))$ for varying values of σ^2 $(x = 1)$. Results were obtained using a Monte Carlo method and the 1D approximation method.

$$k_a = ln(2\pi^{\frac{n}{2}}|\boldsymbol{\Sigma}_a^{-\frac{1}{2}}|) \tag{6.19}$$

$$k_b = ln(2\pi^{\frac{n}{2}}|\boldsymbol{\Sigma}_b^{-\frac{1}{2}}|) \tag{6.20}$$

where $n = 6$, the dimensionality of the feature models.

The Monte Carlo approximation of $\epsilon(p_a(\boldsymbol{v}), p_b(\boldsymbol{v}))$ and $\epsilon(p_b(\boldsymbol{v}), p_a(\boldsymbol{v}))$ is given by:

$$\epsilon(p_a(\boldsymbol{v}), p_b(\boldsymbol{v})) = \frac{1}{n_a}(n_a - \sum_{i=1}^{n_a} c(\boldsymbol{s}_{ai})) \tag{6.21}$$

$$\epsilon(p_b(\boldsymbol{v}), p_a(\boldsymbol{v})) = \frac{1}{n_b}(\sum_{i=1}^{n_b} c(\boldsymbol{s}_{bi})) \tag{6.22}$$

where n_a and n_b are the number of random samples from $p_a(\boldsymbol{v})$ and $p_b(\boldsymbol{v})$ respectively, $\boldsymbol{s}_{ai}$ is the i^{th} random sample from $p_a(\boldsymbol{v})$ and $\boldsymbol{s}_{bi}$ is the i^{th} random sample from $p_b(\boldsymbol{v})$.

Figure 6.3.1 shows the effect on $\epsilon(p_a(\boldsymbol{v}), p_b(\boldsymbol{v}))$ and $\epsilon(p_b(\boldsymbol{v}), p_a(\boldsymbol{v}))$ as x is varied whilst σ^2 remains constant. Figure 6.3.1 shows the reverse situation, σ^2 is varied whilst x remains constant. The Monte Carlo method should give a relatively accurate answer but is to slow to use in practice. The graphs show that the 1D approximation scheme follows the same trends as the Monte Carlo method. This suggests that the 1D scheme can give a satisfactory approximation.

6.3.2 Constructing a Saliency Map

To recap, in order to compute s_k for each feature, the following steps are carried out:

- Approximate each features mean, $\boldsymbol{\mu}_k$, and covariance, $\boldsymbol{\Sigma}_k$, using equations 6.2 and 6.3.
- Estimate $\epsilon(p_k(\boldsymbol{v}), p_j(\boldsymbol{v}))$ using the 1 dimension approximation as shown in Section 6.3.
- Compute s_k using equation 6.4

Once the Saliency measure, s_k, of each feature has been found using equation 6.4, the result can be visualised by constructing a saliency map. As before this is done by taking the mean image of the object and at each pixel plotting the saliency measure corresponding to the most salient feature centered on that pixel. The resulting image indicates which areas of the object are the most salient.

Figure 6.7(b) shows an example of such a saliency image computed from feature models trained on approximately two hundred images of faces. Figure 6.7(a) shows the mean face with the peaks of the saliency image super-imposed.

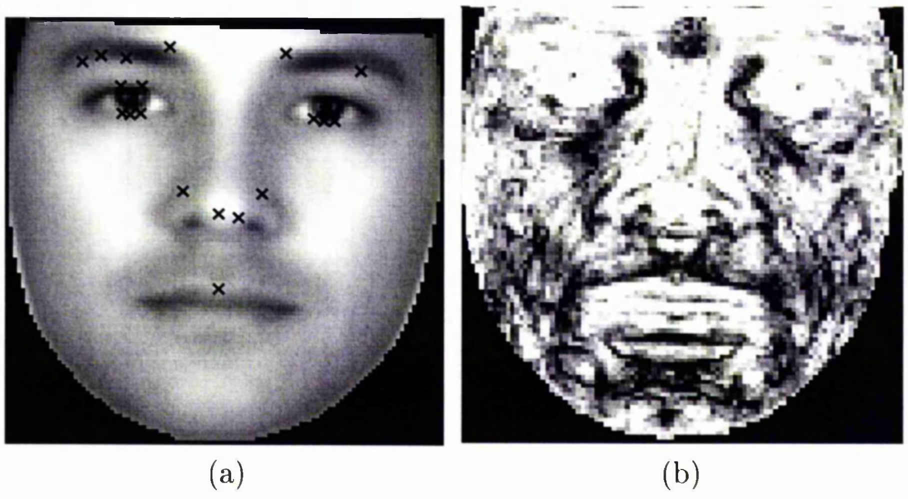


Figure 6.7: (b) is the saliency image obtained from approximately 100 face images, white regions are most salient. (a) show the peaks of the saliency image superimposed onto a 'mean' face

6.4 Results

In order to quantify if the salient features selected using Object Feature Saliency are more useful than those selected using Image Feature Saliency (Chapter 5), we repeated the experiments from Section 5.6 with features selected using Object Feature Saliency.

We selected 20 features by applying Object Feature Saliency to a landmarked set of 100 face images. As in the previous results we attempted to locate these features in a test set of 188 unseen images of faces. We recorded the rank of the correct match for each search. Note that none of the 100 training images were repeated in the 188 test images.

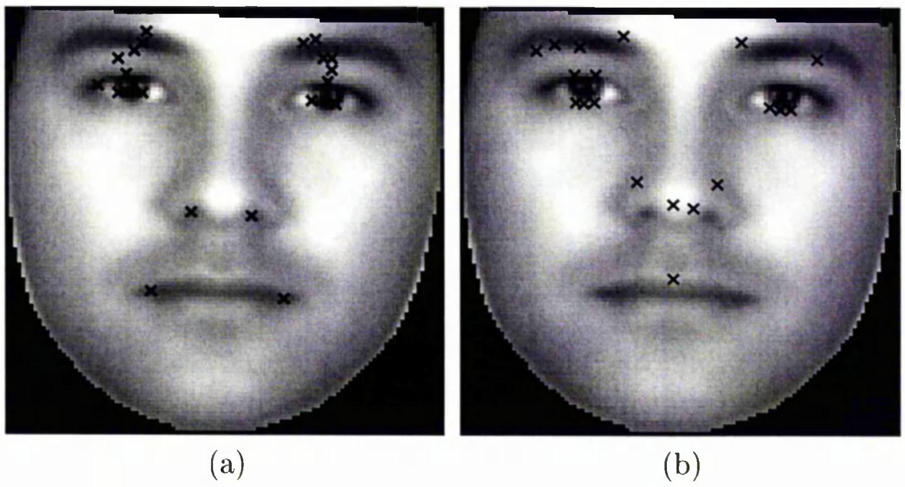


Figure 6.8: Salient features selected using Image Feature Saliency (a) and Object Feature Saliency (b). The Object Feature Saliency method used a training set of 100 images, the Image Feature Saliency used the mean image calculated from the same 100 training images.

Figure 6.8(a) shows the 20 features selected using Object Feature Saliency and Figure 6.8(b) shows 20 features selected using Image Feature Saliency). The Object Feature Saliency method used a training set of 100 images. The Image Feature Saliency method used the mean image calculated from the same 100 training images in order to measure saliency. Note that the Object Feature Saliency method has chosen not to select the corners of the mouth as salient features (unlike the Image Feature Saliency method), possibly because these features vary significantly with expression. Instead the Object Feature Saliency method has selected a feature at the center of the top lip, a feature whose appearance does not vary significantly with pose or expression variations.

Figure 6.9 shows the graph of success rate against false positives. It contains the previous search results from Section 5.6 together with the results obtaining using features selected with Object Feature Saliency. It can be seen the features selected using the Object Feature Saliency metric were found with a greater degree of success than features selected using Image Feature Saliency, by hand or randomly.

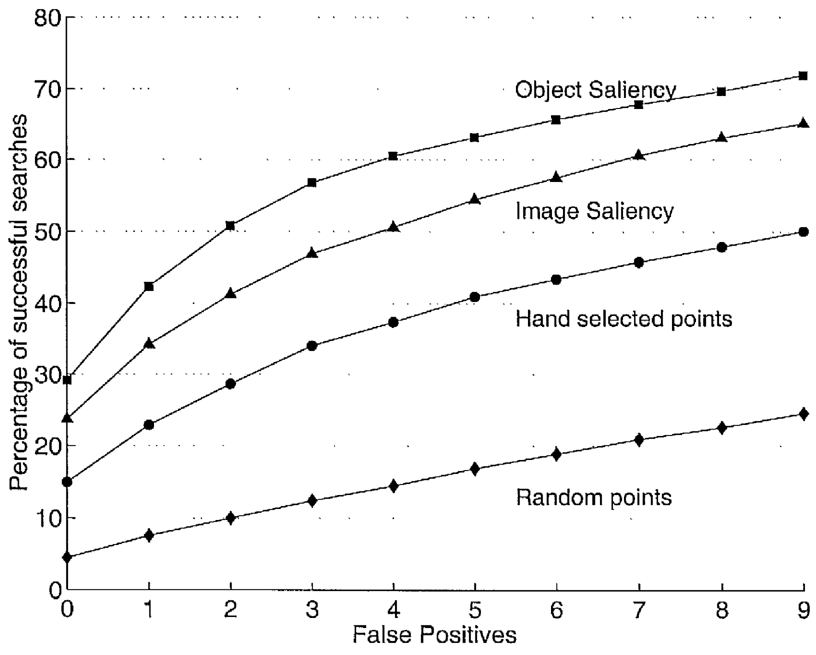


Figure 6.9: Graph illustrating the percentage of successful searches according to the number of false positives. The graph can be interpreted in the same manner as a ROC curve, i.e. the closer the curve is to the top left, the better the result.

6.5 Discussion

We have described how a probabilistic measure of saliency, calculated from a number of object examples, can be used to select those object features least likely to generate false positive matches in unseen images. This metric is known as *Object Feature Saliency.*

We have also shown that the salient features selected using Object Feature Saliency can be found with a greater degree of success in unseen object examples than either features selected using Image Feature Saliency (Chapter 5) or features selected by hand.

Object Feature Saliency is not used in the remainder of the thesis because of its dependence on having pre-defined correspondences between training examples. Despite this we believe it to be a useful metric which could be employed in many feature based systems.

The aim of this thesis is to explore methods of automatically building statistical models of shape and appearance. A central part of this task is obtaining robust correspondences between image examples, the subject of the previous two chapters. In the following chapter we describe an attempt to build models automatically from image sequences, using salient features to find correspondences between frames.

Some of the ideas presented in this chapter were presented at the British Machine Vision Conference 1998 [86].

Chapter 7

Automatic models from Image Sequences

This chapter describes a method for a automatically placing landmarks across an image sequence to define correspondences between frames. The marked up sequence can then be used to build a statistical model of the appearance of the object within the sequence.

This task is seen as a step towards the main goal of this thesis, which is to automatically build models from arbitrary image sets, not just sequences. Even so, being able to automatically build models from image sequences is still a task of great value. In the modern world video sequences are everywhere, in television broadcasts and security footage to name but two. The ability to automatically construct models of the objects featured in this footage would be of great use in the animation and computer graphics industry.

The approach adopted attempts to take advantage of the fact that objects and their features often do not change significantly between frames. This is a constraint which is not available when considering arbitrary image sets. Thus automatic model building from image sequences is a more heavily constrained problem than automatic model building from arbitrary image sets and hence a slightly more tractable one.

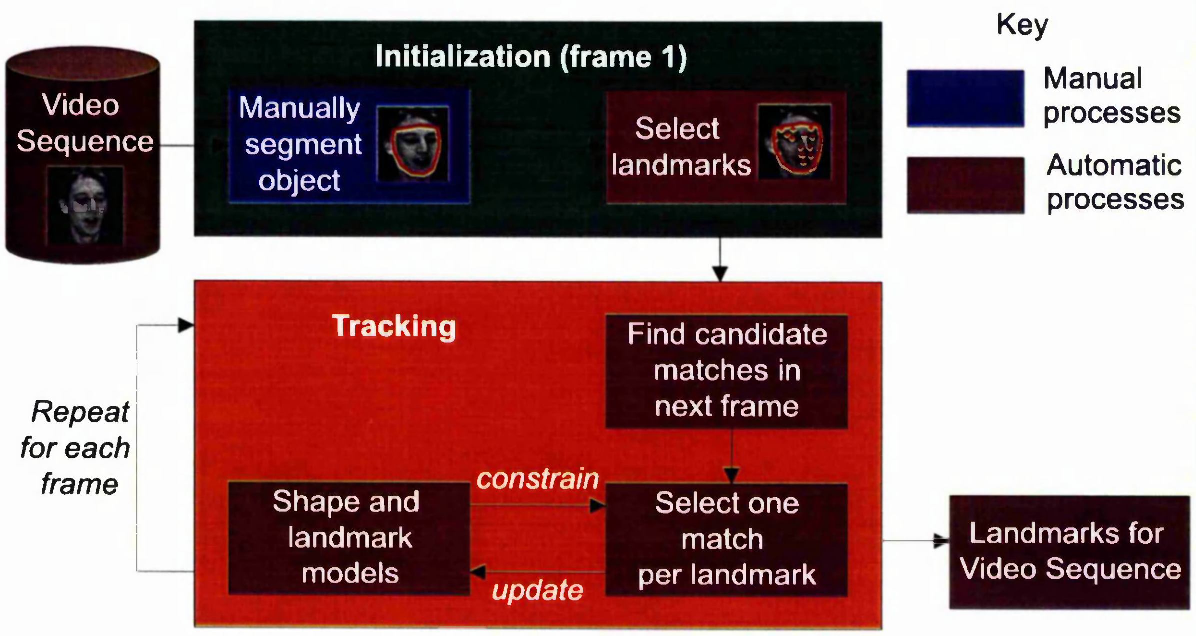


Figure 7.1: An overview of the track scheme

Figure 7.1 illustrates an overview of the method. This consists of two main stages, initialisation and tracking. The aim of the initialisation stage is to decide which features should be selected as landmarks. The tracking stage is then repeated for each subsequent frame in turn, attempting to locate the position of each of the landmarks selected in the initialisation phase in the new frame. As the sequence is searched, we build a statistical model of each landmark feature together with a model of the overall shape of the landmarks. These models can then be used to improve the probability of getting correct matches in subsequent frames.

The following will explain, in detail, the initialisation and tracking stages.

7.1 Initialisation

As shown in Figure 7.1 the initialisation phase consists of two steps, both of which are applied to the first frame of the sequence. The first step is the only one which requires human interaction. It requires the user to manually select a region of the object that they wish to model from the first frame of the sequence. Without this prior information the system has no way of knowing what it is in the sequence the user is interested in modeling. The segmentation need only be approximate, but it is important that the selected region is wholly within the object. The segmented region is used to select features with which to represent the object. The presence of background features would confuse subsequent tracking.

Figure 7.2 illustrates examples of good and bad segmentations.

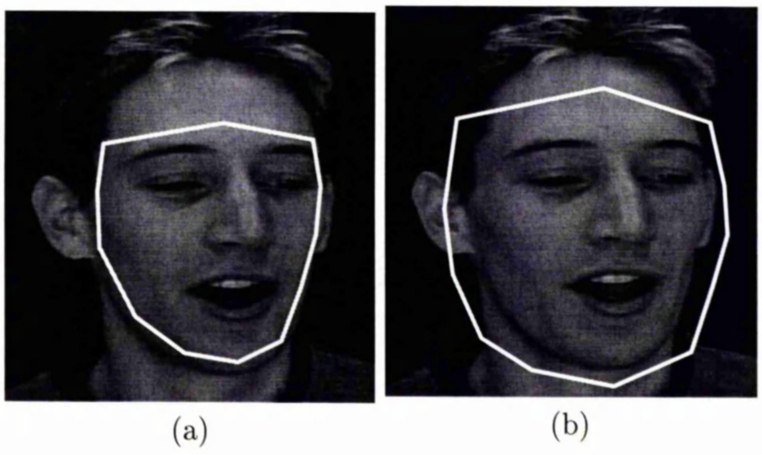


Figure 7.2: Examples of good (a) and bad (b) segmentations. No background should be included.

The final step in the initialisation phase is to select a number of salient features, those which are most likely to be relocated correctly in subsequent frames. Image Feature Saliency (Chapter 5) is applied to the image features within the segmented region at a number of scales in order to select the n most salient features. The image features which lie outside of the segmented region are not included in the saliency calculation.

The result of this analysis is n salient features within the selected region which are described by $\boldsymbol{v}_{1k}$, s_k and $\boldsymbol{x}_{1k}$, where $\boldsymbol{v}_{ik}$ is the feature vector of salient feature k in frame i, s_k is the image pyramid level at which feature k was found to be salient ($s_k = 0$ being the original image and $s_k = 1$ being half the size of the original etc), and $\boldsymbol{x}_{ik} = (x_{ik}, y_{ik})$ are the coordinates of feature k in frame i respectively.

7.2 Tracking: Locating the features in the ith frame

After the initialisation phase, the sequence is processed sequentially (frame by frame) by the tracking phase. In the following sections we describe how having tracked to frame n_f we locate possible matches for each feature in frame $n_f + 1$, how several hypotheses are then generated for the location of the object in frame $n_f + 1$ and finally how shape constraints are applied to select the most probable hypothesis.

7.2.1 Locating feature matches

The following assumptions can be made about the object and its features between frames n_f and $n_f + 1$:

- The features will not move more than a given number, r, pixels between frames.
- The scale of the object and therefore features will not change significantly between frames.
- The appearance of the object and therefore the features do not change significantly between frames.

These assumptions help constrain the search and reduce processing time. Only πr^2 candidate pixels in each frame need to be considered in order to locate the best match for one particular feature.

A candidate pixel from frame $n_f + 1$ has an associated feature vector, $\boldsymbol{v}$. The quality of match, $M_k(\boldsymbol{v})$, of the k^{th} salient feature from frame 1 and the feature vector, $\boldsymbol{v}$, associated with a candidate pixel from frame $n_f + 1$ is defined as:

$$M_k(\boldsymbol{v}) \quad = \quad (\boldsymbol{v} - \bar{\boldsymbol{v}}_k)^T \boldsymbol{S}_k{}^{-1} (\boldsymbol{v} - \bar{\boldsymbol{v}}_k) \tag{7.1}$$

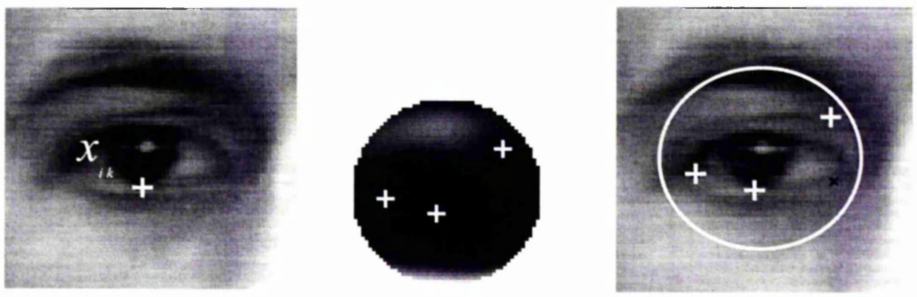
where $\bar{\boldsymbol{v}}_k$ and $\boldsymbol{S}_k$ together are known as the *feature model* of the k^{th} salient feature from frame one. $\bar{\boldsymbol{v}}_k$ and $\boldsymbol{S}_k$ are defined as follows:

$$\bar{\boldsymbol{v}}_k = \frac{1}{n_f} \sum_{i=1}^{n_f} \boldsymbol{v}_{ik} \tag{7.2}$$

$$\boldsymbol{S}_k = \begin{cases} \boldsymbol{C}_{s_k} & \text{if } n_f = 1 \\ \frac{1}{n_f - 1} \left(\sum_{i=1}^{n_f} (\boldsymbol{v}_{ik} - \bar{\boldsymbol{v}}_k)^2 + \lambda_r \boldsymbol{I} \right) & \text{if } n_f > 1 \end{cases} \tag{7.3}$$

where C_σ is the covariance matrix of all the feature vectors extracted from within the object boundary from frame one at pyramid level σ and $\lambda_r \boldsymbol{I}$ is a regularising term to avoid singular matrix inversion. The smaller $M_k(\boldsymbol{v})$, the better the match. $M_k(\boldsymbol{v})$ is linearly related to the log of the probability that $\boldsymbol{v}$ comes from the distribution described by feature model k.

By calculating $M_k(\boldsymbol{v})$ for all candidates for a particular feature, k, a similarity image can be formed. This is done by plotting $M_k(\boldsymbol{v})$ in the image domain of frame $n_f + 1$. We locate the m best matches for each feature by locating the m lowest troughs in its similarity image (small values indicates a good match). Figure 7.3 shows an example of this.



(a) frame i (b) similarity (c) frame $i+1$

Figure 7.3: Calculating a feature match in subsequent frames. (b) is the similarity image obtained whilst searching for the feature in (a) in the frame shown in (c). The troughs in (b) represent the matches, these are highlighted in (c). The circle in (c) represents the region searched.

7.2.2 Forming and evaluating object hypothesis

The shape of the object in a previously searched frame is represented by n salient features. In the previous section m possible candidate matches for the location of each salient feature in frame $n_f + 1$ were found. By selecting one candidate match for each salient feature we form an *object hypothesis*. Generating all possible combinations would result in m^n object hypotheses for frame $n_f + 1$. In order to reduce the number of object hypotheses, candidate matches which correspond to the largest similarity values, $M_k(\boldsymbol{v})$, are discarded (i.e. we remove the most improbable matches). The number of matches discarded depends on the computational power available.

Each hypothesis is given a probability of being correct, based on the evidence obtained from previous frames. The probability of a hypothesis being correct is equal to the probability of each feature being correct together with the probability of the shape formed by the salient features being correct. The shape formed by the n salient features, for a particular frame, i, is described by a $2n$ element *shape vector*, $\boldsymbol{x}_i$, where:

$$\boldsymbol{x}_i = (x_{i1}, \ldots.., x_{in}, y_{i1}, \ldots.., y_{in})^T \tag{7.4}$$

The quality of fit of a shape vector, $\boldsymbol{x}_i$, in frame, $n_f + 1$, is given by:

$$M(\boldsymbol{x}_i) \quad = \quad (\boldsymbol{x}_i - \bar{\boldsymbol{x}})^T \boldsymbol{H}^{-1} (\boldsymbol{x}_i - \bar{\boldsymbol{x}}) \tag{7.5}$$

where $\bar{\boldsymbol{x}}$ and $\boldsymbol{H}$ are the mean and covariance matrix of shape vectors from frame 1 to n_f after they have been aligned using Procustes Analysis [39]. If $n_f = 1$ then $H = \boldsymbol{I}\sigma$ where $\boldsymbol{I}$ is a $2n$ by $2n$ identity matrix and σ is an estimate of the variance of the features between frames in pixels. $M(\boldsymbol{x}_i)$ is also linearly related to the log probability that $\boldsymbol{x}_i$ is drawn from the distribution shape vectors from frames 1 to n_f.

It is assumed that features can be treated independently of one another and the global shape. A measure of the quality of a particular hypothesis, $M(h_k)$, is thus given by:

$$M(h_r) = M(\boldsymbol{x}_{h_r}) + \sum_{k=1}^{n} M_i(\boldsymbol{v}_{h_r k}) \tag{7.6}$$

where h_r is the r^{th} object hypothesis, $\boldsymbol{x}_{h_r}$ is the shape vector for hypothesis h_r

and $v_{h_r k}$ is the feature vector that hypothesis h_r has selected to describe feature k. The hypothesis with the smallest $M(h_r)$ is selected as the most likely solution.

7.3 Results

In order to test the method, four test sequences were processed and the correspondence obtained from each used to build an appearance model. Each test sequence was of a human face reading out a single sentence. The sequences contain both expression and pose variation. Figure 7.4 shows frames from each of the four sequences. These sequences will also be used as test data in the chapters that follow.

Of the four sequences processed, only two of the sequences (sequence A and B) were successfully tracked from beginning to end. Figure 7.5 illustrates the modes of variation for an appearance model trained on the correspondences obtained from a successfully tracked sequence. As a comparison we also built two other models for each of the successfully tracked sequences. The first was trained using 25 hand placed landmarks , the second trained using only 4 landmarks placed on each corner of the smallest rectangle which could contain the face in each frame, essentially an Eigenface model [82]. This second model was used to examine the affect on the texture error if no shape normalisation is used. Figure 7.8 shows the positions for the landmarks for the hand placed model (a) and the automatic model (b). The positions of the landmarks used for the manually trained model were chosen by selecting landmark positions typically used when training appearance models of the face. The Figure 7.6 shows the modes for the manually trained model and Figure 7.7 show the modes for the eigen model. Note that the modes of the manually trained and automatically trained models appear very similar, and considerably 'sharper' than the blurred modes of the eigen model. It is a good initial indication that the automatically generated model is of a similar quality to the manually trained model.

It is difficult to assess the results quantitatively without considering the applica-

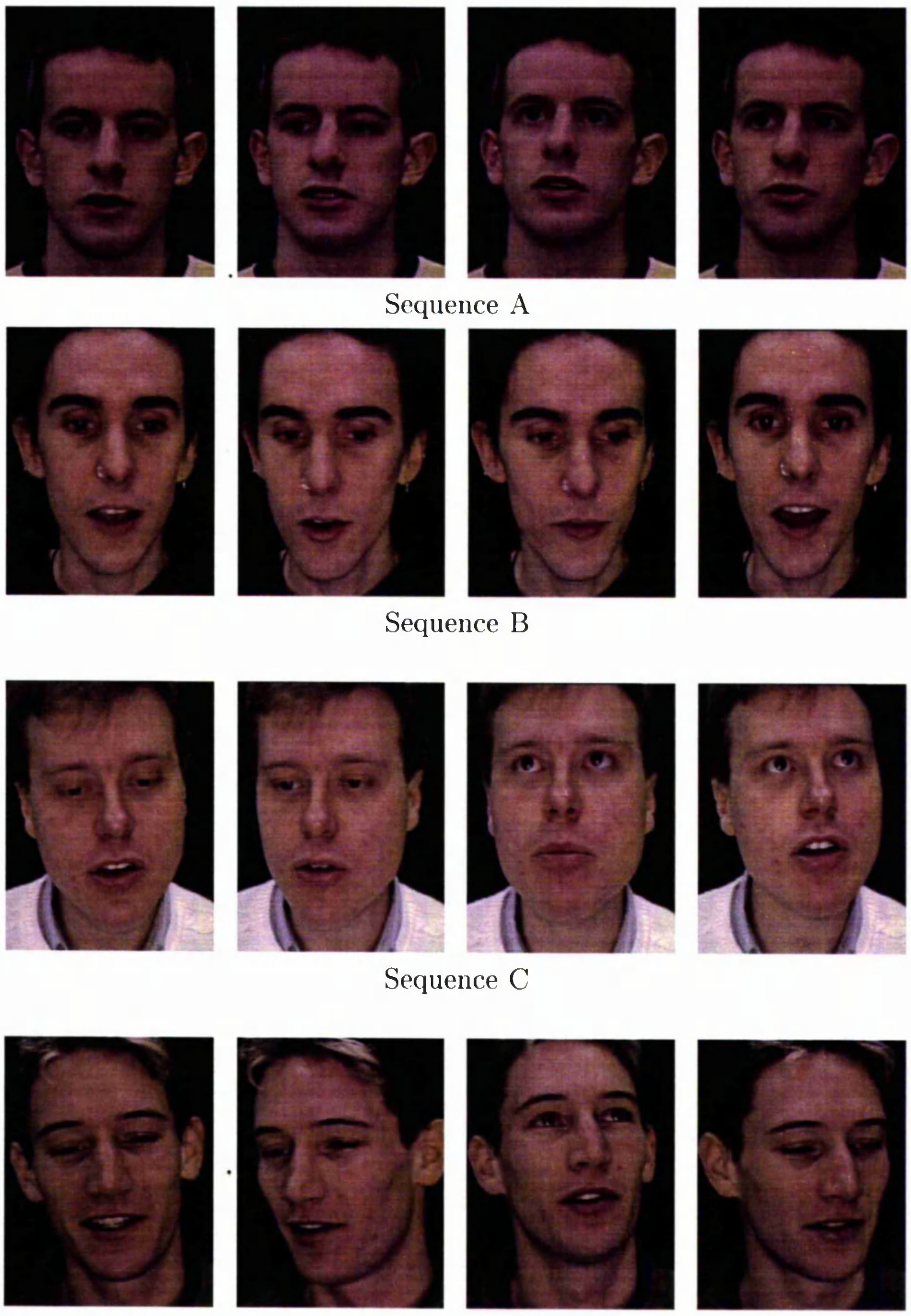


Figure 7.4: Example frames from each of the four sequences used in the results.

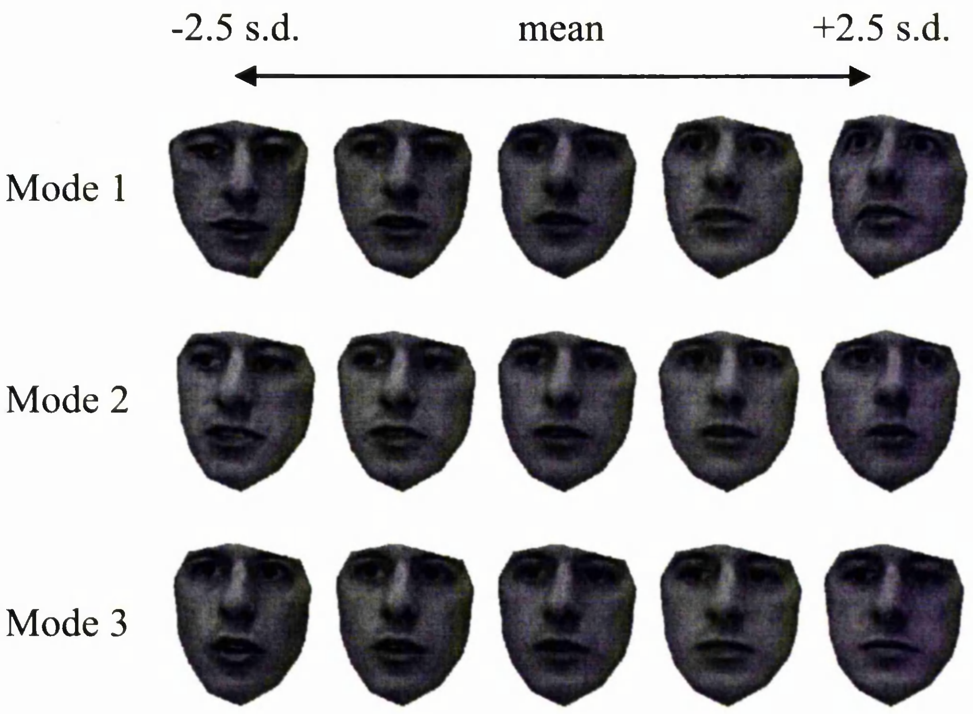


Figure 7.5: The first 3 modes of variation of an Appearance model training using automatically generated landmarks from sequence A.

tion to which the model will be put. For effective coding (compression), we require that truncating the number of models modes has a minimal effect on the reconstruction error. We can thus compare models by how the reconstruction error varies with the number of modes.

For a particular number of model modes we find the best fit between the model and each training image. The texture error is defined as the root mean squared grey level difference between the original image and the model's best fitting reconstruction. The shape error is the root mean squared difference between the original and reconstructed landmark positions in pixels. Figure 7.9 and Figure 7.10 shows how the reconstruction texture error and the shape error decrease as the model modes increase. The graphs show that both the texture error and shape error are reduced by training the model automatically, no matter how many modes of variation are chosen from the model. The automatic method can thus lead to a more compact model. Note that the eigen

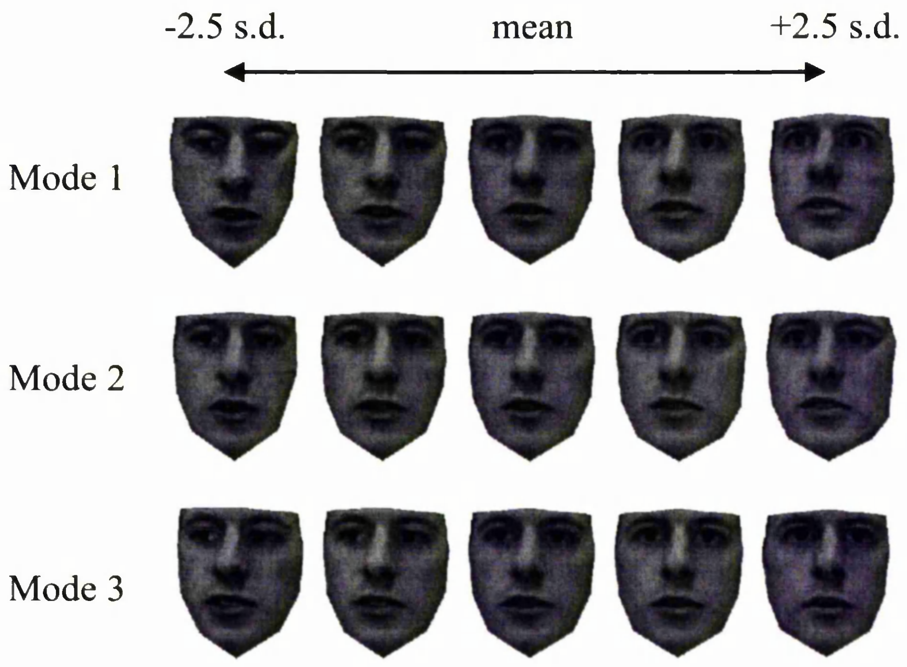


Figure 7.6: The first 3 modes of variation of an Appearance model training using manually placed landmarks from sequence A.

model has the greatest texture error indicating that normalising with a shape model before computing texture also leads to a more compact model.

Examining an animation of the sequence with the landmarks for each method superimposed also suggests the automatic method is preferable to the manual method. The animation of the manually selected landmarks clearly shows the landmarks jumping around erratically, not moving smoothly with the features they where intended to highlight. The automatically selected landmarks moved much more smoothly with the image features. The reason for the noise in the manually trained landmarks is due to human error in the marking up process. This is further evidence to suggest that landmarking image data is a highly error prone task.

An important conclusion drawn from these results is that models built from manually placed landmarks are not a gold standard and can be improved upon using other methods.

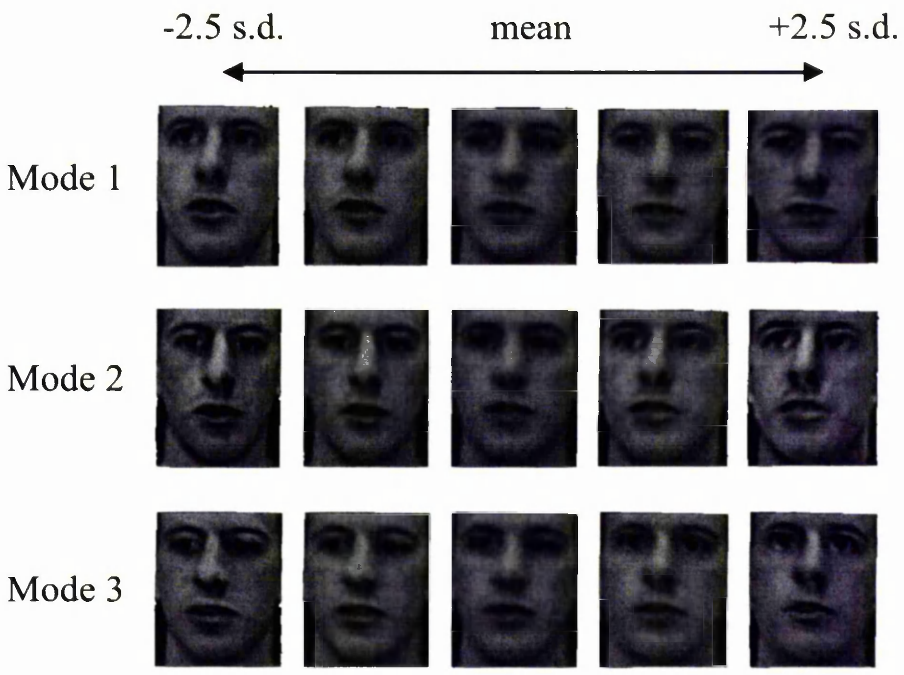


Figure 7.7: The first 3 modes of variation of an Eigenface model from sequence A.

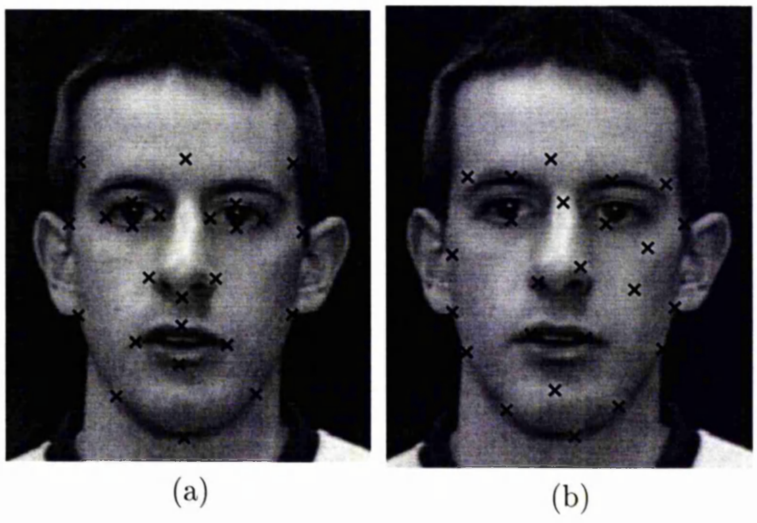


Figure 7.8: Landmarks chosen manually (a) and automatically (b).

7.4 Discussion

We have demonstrated one possible approach to automatically training appearance models. The system both locates the most suitable features to use as model points,

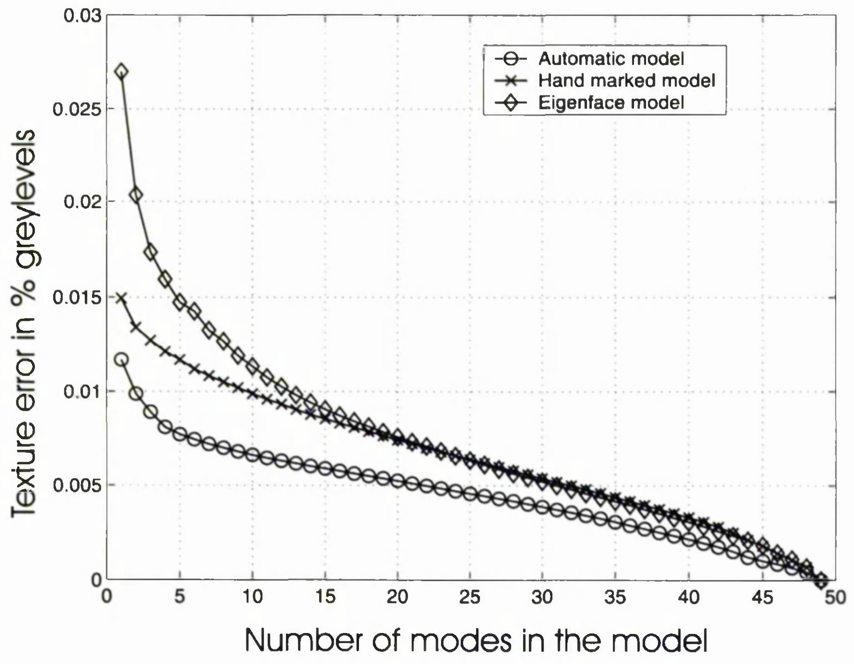


Figure 7.9: Compares the texture error of an automatically built model with that of an Eigenface model and a model trained using hand placed landmarks.

and tracks their correspondence across frames. We have shown that the method can be used to make models of the human face for image sequences automatically.

In long sequences, features which were salient in the first frame are not necessarily still salient in the 50th frame. For example, a salient feature may regularly become obscured. One possible solution to this would be to incorporate object saliency (Chapter 6). When searching frame $n_f + 1$, instead of locating features which were salient in frame 1, we could locate features which were salient in frames n_f to, say, $n_f - 5$. However, this would require additional work (eg using interpolation between points) to compute correspondences across the entire sequence.

The results presented in this chapter have used a relatively small number of landmarks (25). We have observed that the coarse scale features are, in general, much more reliable than the fine scale features. The scheme could be extended to find a greater number of correspondences by adopting a multi-scale approach. Coarse scale

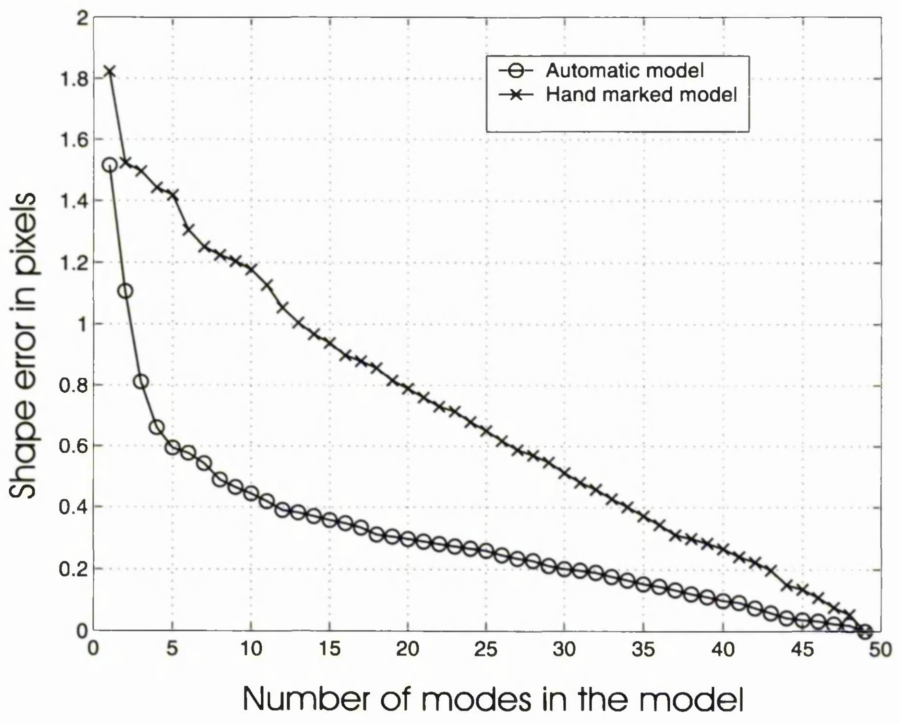


Figure 7.10: Compares the shape error of an automatically built model with that of a model trained using hand placed landmarks.

salient features would be found first, then the resulting coarse scale correspondence could be used to constrain a search for finer scale salient features. This would lead to a more accurate overall correspondence.

One of the most encouraging conclusions from this chapter is that automatic methods of model building can result in models which are more compact than models training using manual methods. Models built using manually selected landmarks should not necessarily be viewed as a gold standard, as they may include a significant amount of human error.

However, if features move and change shape significantly between frames, false correspondences can result. Because it is not possible to determine whether a correspondence is false the feature models can get corrupted. This is particularly damaging in the first few frames when the feature and shape models are only trained on relatively few examples, so one bad example has a great effect. Once the models have become corrupt there is little chance of the tracker recovering in subsequent frames. This happened in the 2 sequences which could not be tracked correctly. These problems result because the scheme is a serial one. The improvements suggested above would help, but not overcome these problems. In order to overcome these problems we need to explore a new underlying framework, a *parallel* framework. This will be the subject of the following chapter.

Some of the ideas presented in this chapter were presented at the British Machine Vision Conference 1999 [87].

Chapter 8

Automatic Models from Image Sets

The previous chapter demonstrated that it is possible to build models from image sequences automatically, although not reliably. The reason for failure was identified as a break down in the tracking. Because the problem was tackled in a serial (frame by frame) nature and the feature models were trained on the fly, a bad correspondence between a pair of frames would corrupt the shape and feature models seriously jeopardising the chances of tracking subsequent frames.

From these observations it seemed that a serial approach to automatic model building would never lead to robust results. This chapter looks at a new method which finds a correspondence between all image pairs. This is shown to make the system more robust to bad correspondences as there is no single chain which can be broken. The approach assumes no prior ordering of images in the set, so it is more generally applicable than relying on sequences of images.

8.1 Overview

This chapter presents a method which, given a rough segmentation of the object in each training image, automatically returns a set of correspondences across the entire training set. The correspondences found can then be used to build appearance models of the object. The approach is first to find correspondences between pairs of images, then to use an iterative scheme to estimate consistent correspondences across the whole training set.

A set of salient points are found within each image, I_i, then the best corresponding points in each of the other images, I_j, are found. The set of points in I_i and correspondences in I_j define a mapping, T_{ji}, from $I_i \rightarrow I_j$(Thin plate splines are used to obtain a continuous mapping). However, because we may use a different set of points to find the mapping, $I_j \rightarrow I_i$, T_{ij}, there is no guarantee that $T_{ji} = T_{ij}^{-1}$. In general, for three images $T_{kj}(T_{ji}) \neq T_{ki}$.

We seek to derive a set of new transformations between images, G_{ij}, which are *globally consistent*, ie $G_{ji} = G_{ij}^{-1}$ and $G_{kj}(G_{ji}) = G_{ki}$.

In practice the transformations are represented using the nodes of a grid. Placing this grid in every image creates a mapping from any image to another and back, hence, defining a globally consistent transform. The only errors being caused by the interpolation, or if the grid 'folds'. The new transformations, G_{ji}, should be as close as possible to those derived from the correspondences T_{ji}. An iterative scheme is presented in which the correspondences are used to drive the grid points towards a global solution.

The following sections describe how salient features are used to find robust correspondences between image pairs, and how thin plate splines can then be used to define a continuous transformation (although globally inconsistent), T_{ji}, between image pairs. Then there is an explanation of how an iterative scheme can be employed to calculate a new transformation, G_{ij}, which is globally consistent across the entire training set. This globally consistent transform provides the required dense correspondences necessary to build an appearance model.

8.1.1 Locating correspondences between image pairs

The aim is locate a set of points in each image, I_i, and find the best corresponding points in all other training images, I_j. Correspondences can be located by selecting features in image I_i and locating them in image I_j.

Image saliency (Chapter 5) is applied to each of the training images in order to locate salient features in each image. Selecting salient features in image I_i increases the probability of calculating correct correspondences in image I_j.

In order to select salient features in each image the object must be roughly segmented so that salient background features are not selected. At present this segmentation process is done manually by placing a bounding box roughly around each object.

The result of this saliency analysis is a set of salient features for each training image. Note that the salient features in one training image are likely to be different from the salient features in another training image. $\boldsymbol{x}_{ik}$ describes the spatial position and s_{ik} the scale of the kth salient feature selected from image I_i. $\boldsymbol{C}_{is}$ is the covariance matrix calculated from all feature vectors extracted from training image I_i at scale s.

In order to locate a correspondence between pairs of images we need to locate the best match for each salient feature from image I_i in image I_j. In order to simplify this problem we make the following assumptions about faces:

- The object'sfeatures will not move more than a given number, r, pixels between training examples, relative to their bounding box.
- The scale and orientation of the object and its features will not change signifi-

cantly.

These assumptions help constrain the search and reduce processing time. Only πr^2 candidate pixels in a particular training image need to be considered in order to locate the best match for one particular salient feature. A candidate pixel has an associated feature vector, $\boldsymbol{v}$. The similarity between the k^{th} salient feature from image I_i, and a candidate vector, $\boldsymbol{v}$, is given by:

$$\delta(\boldsymbol{v}_{ik}, \boldsymbol{v}) = (\boldsymbol{v} - \boldsymbol{v}_{ik})^T \boldsymbol{C}_{is}^{-1} (\boldsymbol{v} - \boldsymbol{v}_{ik}) \tag{8.1}$$

where $\boldsymbol{v}_{ik}$ is the feature vector describing the k^{th} salient feature in image I_i. The smaller $\delta(\boldsymbol{v}_{ik}, \boldsymbol{v})$, the more likely a match.

Figure 8.1 illustrates how a match for a salient feature from training image I_i is located in a second training image I_j.

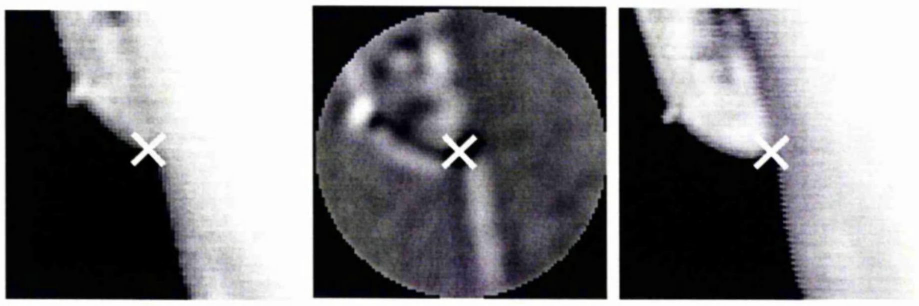
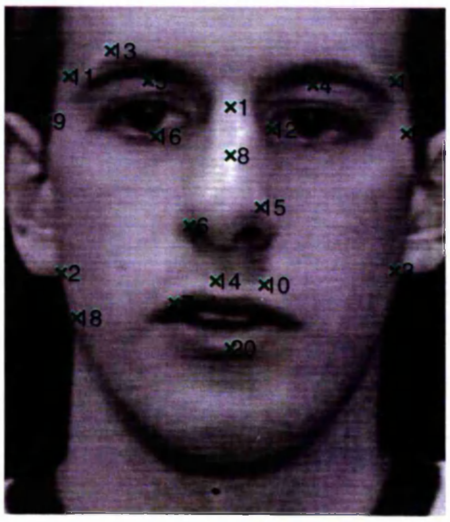

(a) image i (b) similarity (c) image j

Figure 8.1: Calculating a correspondence between image I_i, (a), and I_j, (c). (b) is the similarity image obtained whilst searching for the salient feature from image I_i in image I_j.

By calculating the similarity, δ, for all candidates in image I_j, we can form a similarity image as shown in figure 8.1(b). We locate the best match by locating the lowest trough in the similarity image. Let $\boldsymbol{m}_{ijk}$ be the spatial position in image I_j of the best match to the kth salient feature from image I_i. Let d_{ijk} be the similarity value, δ, of the match $\boldsymbol{m}_{ijk}$. Note that d_{ijk} is also linearly related to the log probability of the match $\boldsymbol{m}_{ijk}$.

8.1.2 Calculating the spatial errors of the matches

For each salient feature, s_{ik}, in image I_i we have located its position, $\boldsymbol{m}_{ijk}$, in image I_j and also a measure of probability, d_{ijk}, of the match being correct. The similarity image as shown in figure 8.1(b) contains further information regarding the errors in the spatial position of the match. Section 8.1.3 shows how this information can be used when calculating the pair-wise image transforms.


(a) image i

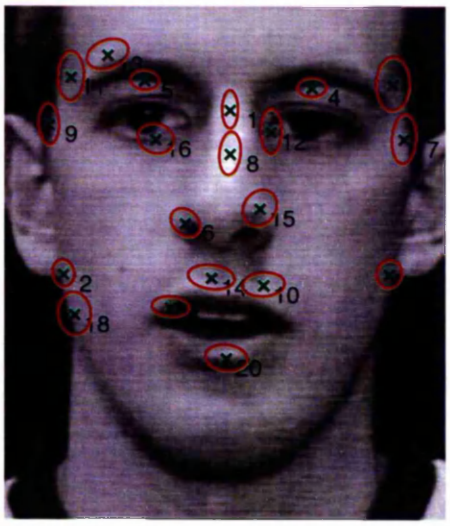

(b) image j

Figure 8.2: The spatial errors associated with salient feature matches. (a) is a training image with its salient features marked. (b) shows a second training image with the position of the best matches to the salient features in (a) shown. The ellipses centered on each match in (b) represent the spatial errors of that match.

For each correspondence we calculate a 2D covariance matrix which describes the spatial errors for the match. The covariance matrix is obtained by fitting a quadratic to the surface [67] of the similarity image.

A quadratic surface which fits a set of data points (x_i, y_i, z_i) is defined by

$$ax_i^2 + bx_iy_i + cy_i^2 = z_i - z_0 \tag{8.2}$$

where a, b and c are constants defining the surface and z_0 is the minimum z value in the data points. Given a set of data points (x_i, y_i, z_i) (sampled from around the best match in the similarity image) the quadratics constants a, b and c can be approximated by solving the following linear system

$$\boldsymbol{X}\begin{pmatrix} a \\ b \\ c \end{pmatrix} = \boldsymbol{Z} \tag{8.3}$$

where

$$\boldsymbol{X} = \begin{pmatrix} x_1^2 & x_1y_1 & y_1^2 \\ x_2^2 & x_2y_2 & y_2^2 \\ . & . & . \\ . & . & . \\ . & . & . \\ x_n^2 & x_ny_n & y_n^2 \end{pmatrix} \qquad \boldsymbol{Z} = \begin{pmatrix} z_1 - z_0 \\ z_2 - z_0 \\ . \\ . \\ . \\ z_n - z_0 \end{pmatrix} \tag{8.4}$$

The covariance matrix which represents the spatial error of a match, $\boldsymbol{S}_m$ is given by

$$ax_i^2 + bx_iy_i + cy_i^2 = \begin{pmatrix} x_i & y_i \end{pmatrix} \begin{pmatrix} a & \frac{1}{2}b \\ \frac{1}{2}b & c \end{pmatrix} \begin{pmatrix} x_i \\ y_i \end{pmatrix} = \boldsymbol{x}_i^T \boldsymbol{S}_m^{-1} \boldsymbol{x}_i \tag{8.5}$$

By analogy with Mahalanobis distance

$$\boldsymbol{S}_m^{-1} = \begin{pmatrix} a & \frac{1}{2}b \\ \frac{1}{2}b & c \end{pmatrix} \tag{8.6}$$

thus

$$\boldsymbol{S}_m = \begin{pmatrix} c & -\frac{1}{2}b \\ -\frac{1}{2}b & a \end{pmatrix} / \begin{pmatrix} ac & -\frac{1}{4}b^2 \end{pmatrix} \tag{8.7}$$

Figure 8.2 shows two training images. The salient features found in the left hand training image are shown and the positions of there matches in the second image are indicated on the right. The ellipses centered on each match represent the spatial errors of that match. Note that features which lie on edges or ridges have a small error perpendicular to the edge but a large error parallel to the edge. This indicates that the match should only be used to constrain a correspondence strongly in a direction perpendicular to the edge.

8.1.3 Transformation between images

Let $\boldsymbol{x}_j = f_{ij}(\boldsymbol{x}_i|\boldsymbol{y}_i, \boldsymbol{y}_j)$ be a mapping from points $\boldsymbol{x}_i$ in image I_i to points $\boldsymbol{x}_j$ in image I_j, controlled by a set of control points $\boldsymbol{y}_i$ and $\boldsymbol{y}_j$. For instance, we can use a thin-plate spline (see Appendix A) with control points $\boldsymbol{y}_i$ and $\boldsymbol{y}_j$ to define this mapping.

The correspondences and associated spatial errors, can be used to define the continuous transformation, T_{ji}, between image I_i and I_j. Rohr *et al* [70] showed how anisotropic control point errors, such as those described above, could be integrated into a thin-plate spline [8]. This means that matches will only constrain the spline strongly where the match was accurately located. Thus

$$T_{ji}(\boldsymbol{x}_i) = f_{ji}(\boldsymbol{x}_i|\boldsymbol{s}_i, \boldsymbol{m}_{ij}) \tag{8.8}$$

where $\boldsymbol{s}_i$ are the salient features in image i, and $\boldsymbol{m}_{ij}$ the corresponding matches in image j. Figure 8.3 shows how the transformation T_{ji} can be used to map a grid in image I_i onto image I_j.

8.1.4 Calculating the Global Correspondence

So far a transformation, T_{ji}, between image I_i and image I_j has been created. This transformation can be computed for all image pairs. However, there is no guarantee that $T_{ji} = T_{ij}^{-1}$ since different correspondence pairs may be used. In general, for three images $T_{kj}(T_{ji}) \neq T_{ki}$. We seek to derive a set of new transformations between images, G_{ij} which are globally consistent, ie $G_{ji} = G_{ij}^{-1}$ and $G_{kj}(G_{ji}) = G_{ki}$.

G_{ji} is represented using the nodes of a grid. Placing this grid on every image

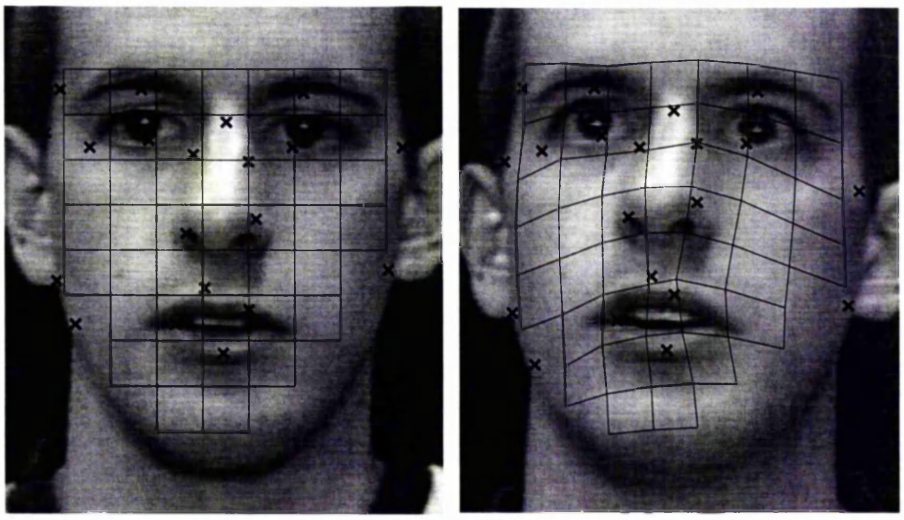


Figure 8.3: Transformation T_{ji} has been applied to the grid in the left hand image in order to locate its position in the right hand image.

allows us to map from any image to another and back.

In order to calculate a globally consistent correspondence across the entire training set we employ an iterative scheme which uses T_{ji} to refine G_{ij} by adjusting the position of the grid nodes.

The scheme is first initialised with an approximation to the final global correspondence. A rectangular grid is placed over each training image. The same grid is placed in all images but is scaled to the approximate size of the target in each image. $\boldsymbol{x}_i$ is the position of the grid nodes in image I_i. This grid is called the *initialisation grid.* Figure 8.4(a) shows an example of this approximate correspondence for a small training set. Note that if we choose to use $\boldsymbol{x}_i$ as landmarks at this stage, the resulting appearance model would be equivalent to an eigen model [82], since no shape deformation is included.

One iteration consists of updating each set of grid points $\boldsymbol{x}_i$ in turn as follows. The pair-wise transformations T_{ji} are used to project every $\boldsymbol{x}_j$ onto the i^{th} image, from which we compute a weighted average. More explicitly, $\boldsymbol{x}_i$ is updated using:

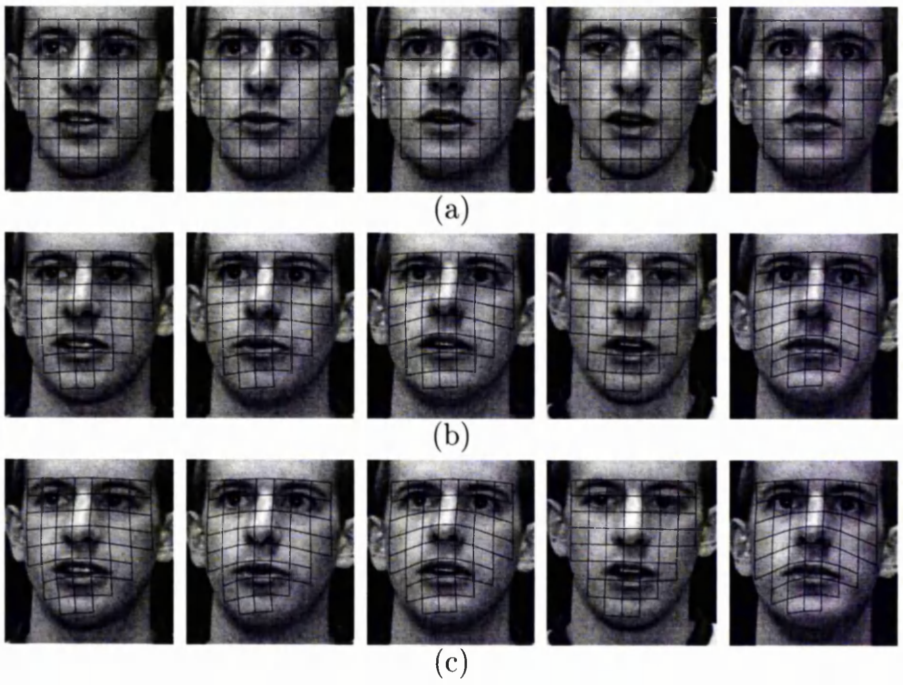


Figure 8.4: Example of how the grid deforms to represent the underlying shape. (a) shows how the grid is initialised, (b) after one iteration and (c) after convergence.

$$\boldsymbol{x}'_i = g(\{\boldsymbol{x}_i\}, \{T_{ji}\}) = (\sum^{n_t} \boldsymbol{W}_j)^{-1} \sum_{j=1}^{n_t} \boldsymbol{W}_j T_{ij}(\boldsymbol{x}_j) \tag{8.9}$$

where n_t is the number of training images and $\boldsymbol{W}_j$ is a diagonal matrix of weights, the p^{th} element of which describes the confidence in the prediction of the position of the p^{th} node of $\boldsymbol{x}_i = T_{ij}(\boldsymbol{x}_j)$. If m_{jik} is the closest salient feature match to the p^{th} node of $\boldsymbol{x}_i$ then the confidence in the prediction is equal to $2\pi^{-\frac{n}{2}}|\boldsymbol{C}_{is}^{-\frac{1}{2}}|exp(-\frac{1}{2}d_{jik})$.

After each iteration it is important to normalise the grid, $\boldsymbol{x}_i$. This is because it is possible for the grid to move off the object slightly. If this is allowed to continue for a number of iterations the effect is magnified. In order to normalise $\boldsymbol{x}_i$ we repropagate the initialisation grid from the first training example, $\boldsymbol{r}$, onto all other examples, using the current $\boldsymbol{x}_i$ as the control points for the mapping. $\boldsymbol{x}_i$ then becomes this newly propagated grid. Mathematically, $\boldsymbol{x}_i$ is normalised as follows:

$$\boldsymbol{x}_i \leftarrow f_{i1}(\boldsymbol{r}|\boldsymbol{x}_1, \boldsymbol{x}_i) \tag{8.10}$$

where $\boldsymbol{r}$ is the initialisation grid for training example 1.

The whole process converges after a few iterations. Figure 8.4 shows how the $\boldsymbol{x}_i$ deforms after each iteration. After one iteration the grids are no longer rectangular, and therefore an Appearance Model built from the grid nodes would capture some shape variation. This will make the model more compact than an equivalent eigen-model.

8.2 Results

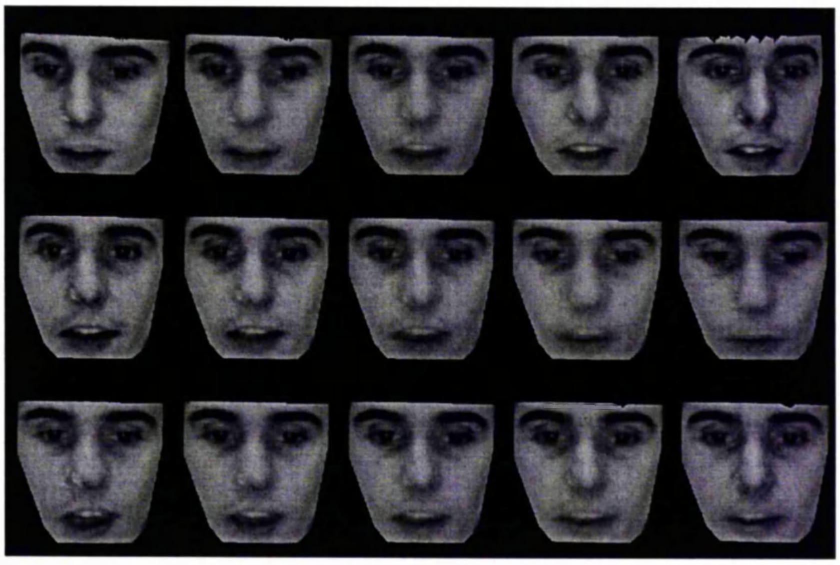


Figure 8.5: Modes of variation for a model trained on automatically generated correspondences.

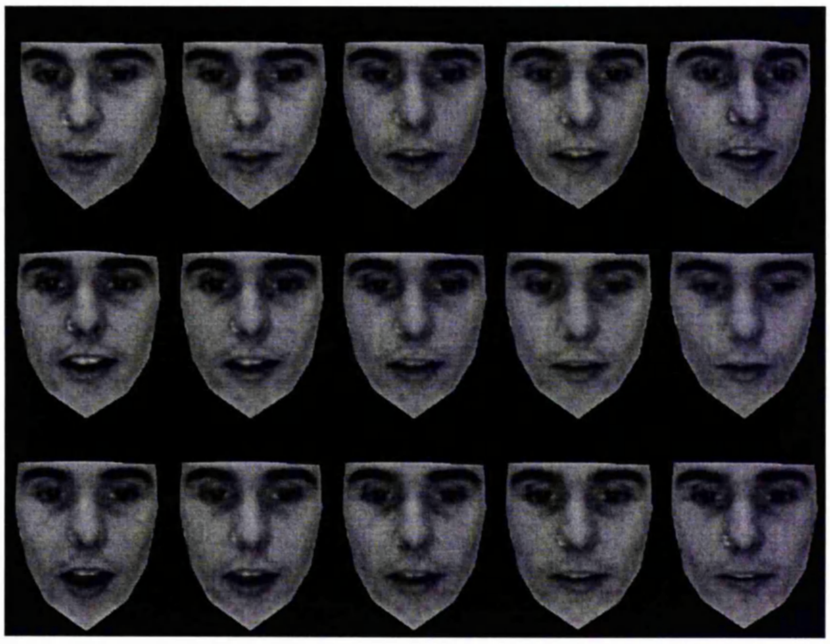


Figure 8.6: Modes of variation for a model trained on manually placed correspondences.

This scheme was used in an attempt to build models automatically from the image sequences used in Section 7.3. One disadvantage of this approach in comparison to the tracking framework (see Chapter 7) is its computational complexity. The tracking framework only required correspondences to be calculated between neighboring frames, whereas the framework presented in this chapter calculates correspondences between all image pairs. More explicitly, if n_t is the number of training images, the computational complexity of the tracking framework is of order n_t compared with n_t^2 for the pair-wise scheme. For this reason only 30 of the 50 frames were used from each sequence.

Figure 8.5 illustrates the modes of variation of a model trained using automatically generated correspondences, and Figure 8.6 shows the modes of a model trained using manually placed correspondences. Note that the modes for each model are of a similar quality. Figure 8.7 shows the modes of an Eigenface model, the modes are much more

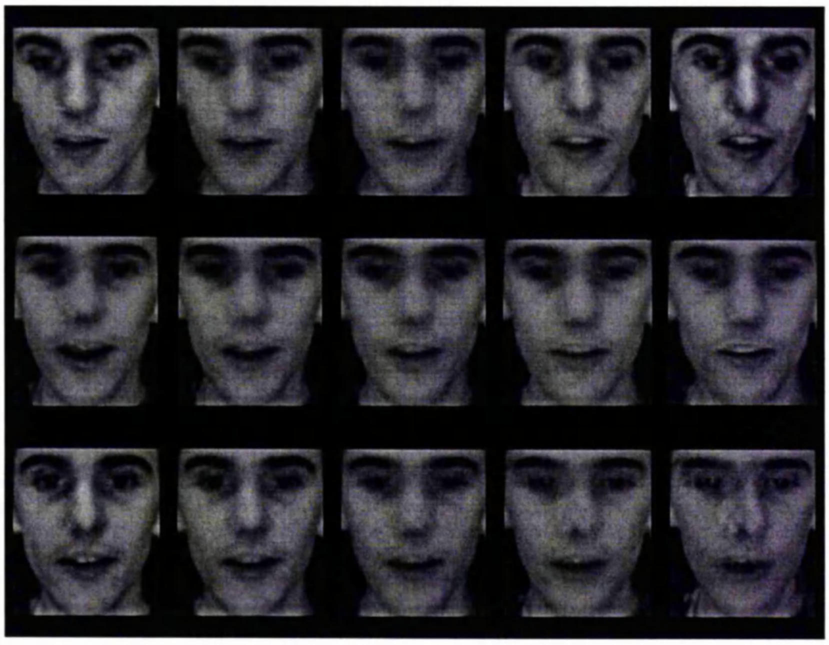


Figure 8.7: Modes of variation for a Eigenface model of the same data as models shown in Figures 8.5 and 8.6.

blurred because no attempt is made to normalise the training images with respect to shape.

We analysed the compactness of the models using the techniques described in Section 7.3. That is we attempted to use the model to reconstruct the training images, measuring the texture and shape error for increasing numbers of model modes. The shape error describes how well the reconstruction predicts the landmarks positions. The texture error describes the model's ability to reconstruct the grey level value for each pixel.

Figures 8.8, 8.9, 8.10 and 8.11 show how the reconstruction texture error, (a), and the shape error, (b), increase as the model modes decrease for each of the sequences mentioned in Section 7.3. For a training set of 30 images, 29 modes reconstructs the training images exactly. As the number of modes is reduced the texture and shape reconstruction error increase. A compact model is one whose curve passes close to the origin. The graphs show that both the texture error and shape error are generally reduced by training the model automatically. The automatic method thus leads to a more compact model in general for these training sets. The texture error for sequence D (Figure 8.11(a)), was of a similar quality the manually trained model. The only case where the reconstruction error was worse than the manually trained model was the texture error of sequence C (Figure 8.10(a)).

In summary, out of the 4 training sets tested, the automatically trained model was more compact for 2 of the training sets (sequences A and B) and for the other 2 sets (sequences C and D) the automatic and manually trained models were of similar or worse quality. All models were an improvement over the Eigenface models.

8.3 Discussion

This chapter has developed a new parallel approach to automatically training appearance models. The system calculates globally inconsistent transformations between all image pairs. These transformations are used to drive an iterative scheme which calculates a globally consistent set of transforms. The globally consistent set of transforms are used to provide the correspondence necessary to build an appearance model.

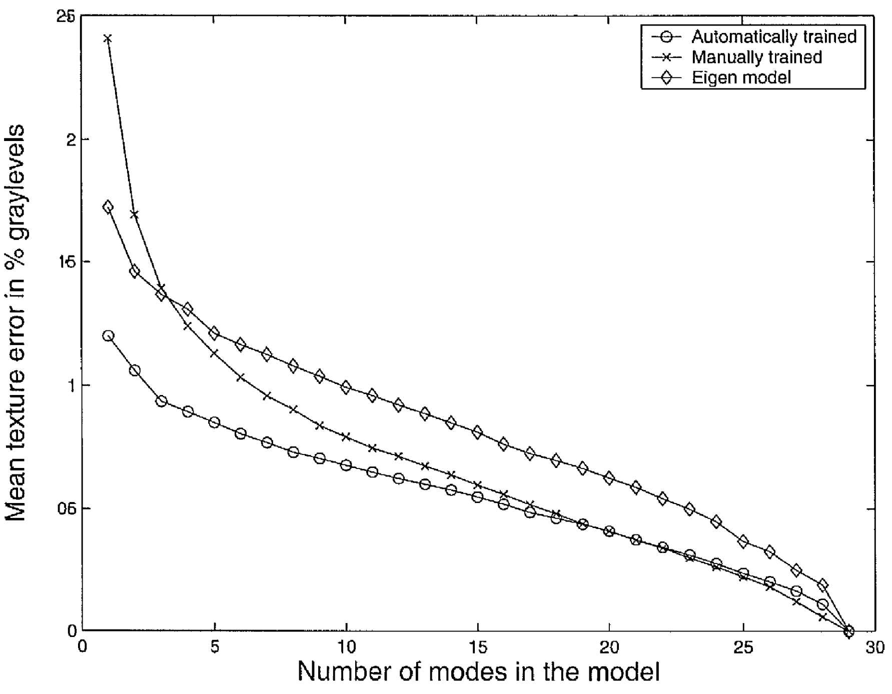
This work should be considered as a general method in which to obtain globally consistent transforms from pair-wise transforms. The pair-wise transforms could be generated from other methods such as optical flow.

The results showed that for the 4 training sets, 2 (sequences A and B) resulted in models which where more compact than the manually trained models and 2 (sequences C and D) resulted in models which were of a similar or worst quality. All models were an improvement over the Eigenface models.

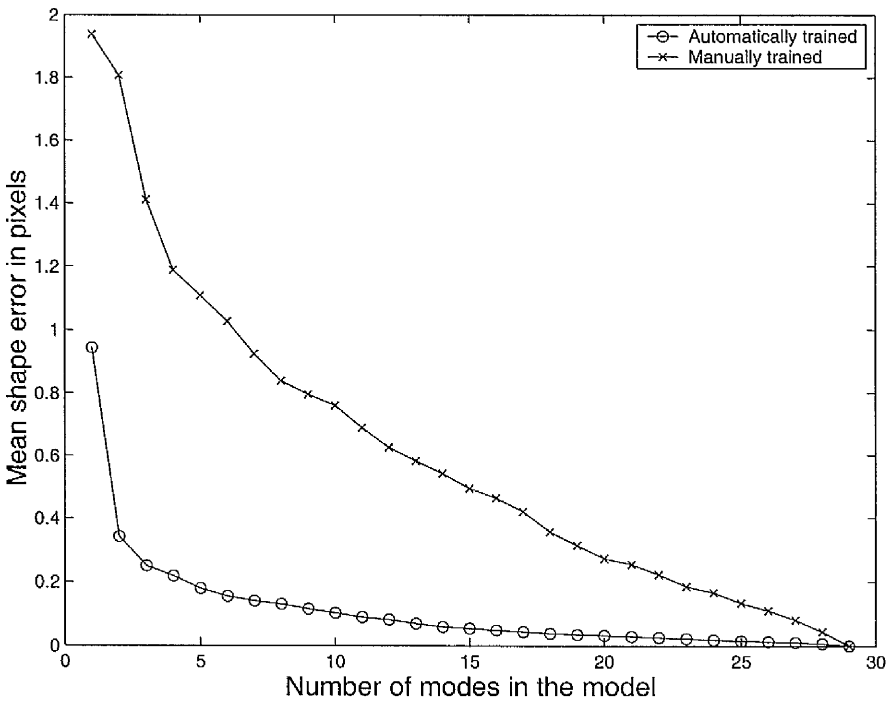
All the training sets resulted in a reasonable model, unlike the tracking scheme. The tracking scheme is a serial scheme which is susceptible to failures when bad correspondences are found early on in the tracking process. The parallel scheme presented in this chapter is much more robust to bad correspondences. This is because correspondences are found between all pairs of images, so bad ones (ones with large errors) are given less significance.

Sequence C and D had a large amount of pose variation in comparison to sequences A and B and this is thought to be the reason why the resulting automatic texture models did not out-perform the manual texture models. More explicitly sequences C and D did not obey the first assumption described in Section 8.1.1. This is a problem which is addressed in the following chapter.

Some of the ideas presented in this chapter were presented at the International Conference on Automatic Face and Gesture Recognition 2000 [90].

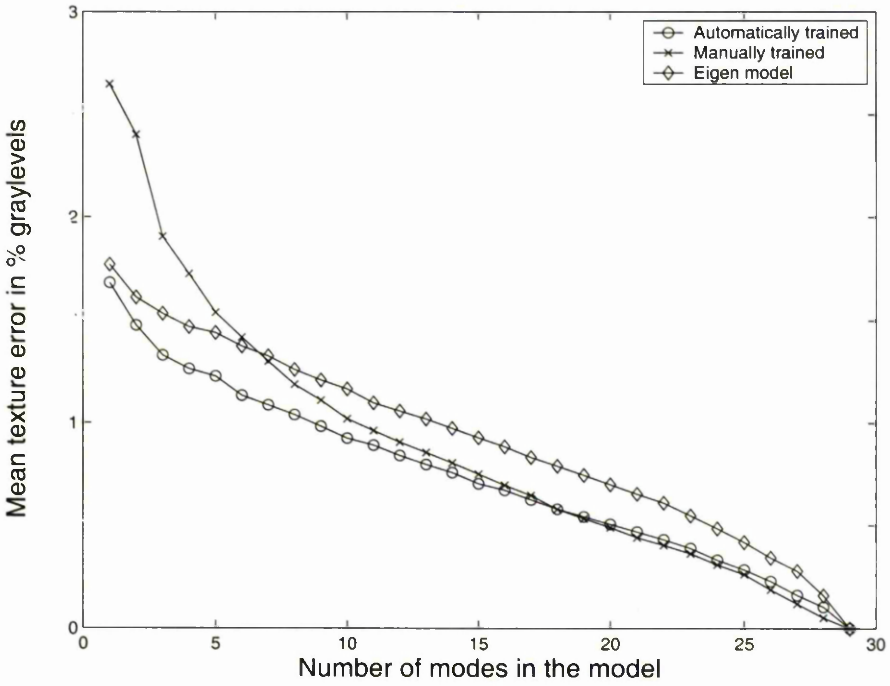


(a)

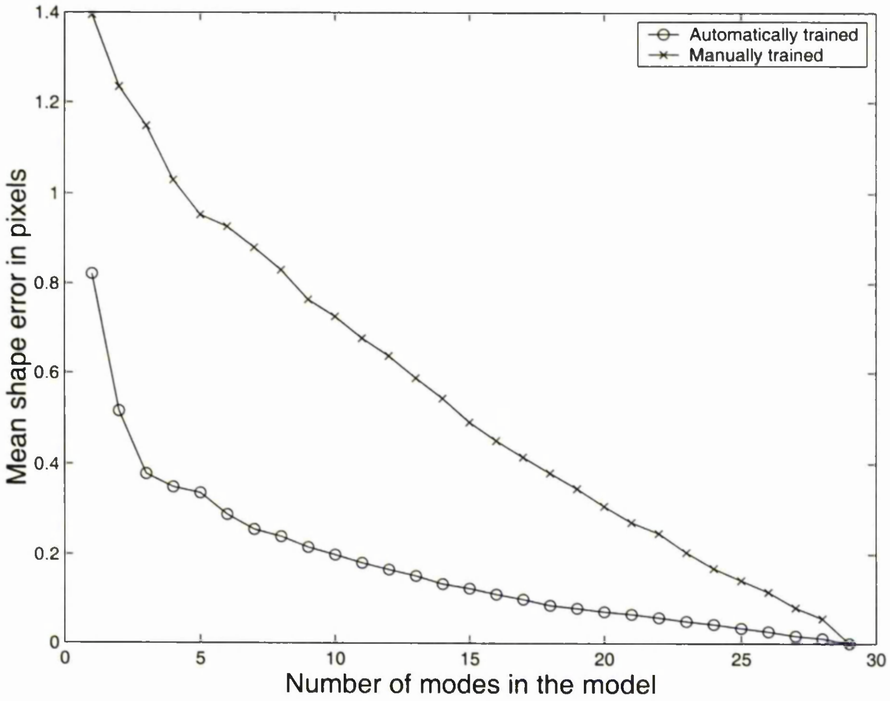


(b)

Figure 8.8: Compares the texture error (a) and shape error (b) of an automatically built model with that of an Eigenface model and a model trained from hand placed landmarks. The models were build using 30 frames of sequence A (Section 7.4).

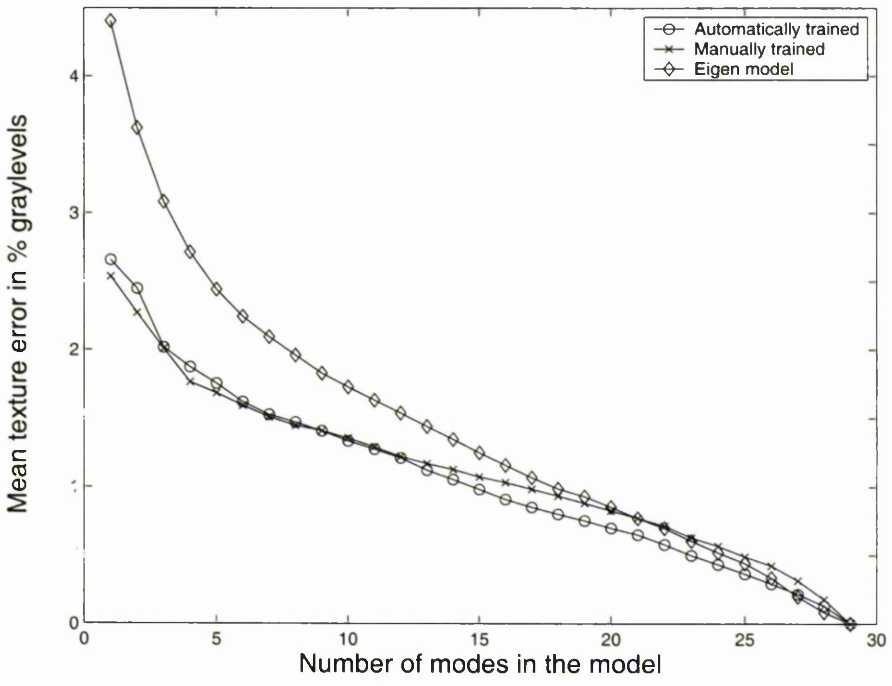


(a)

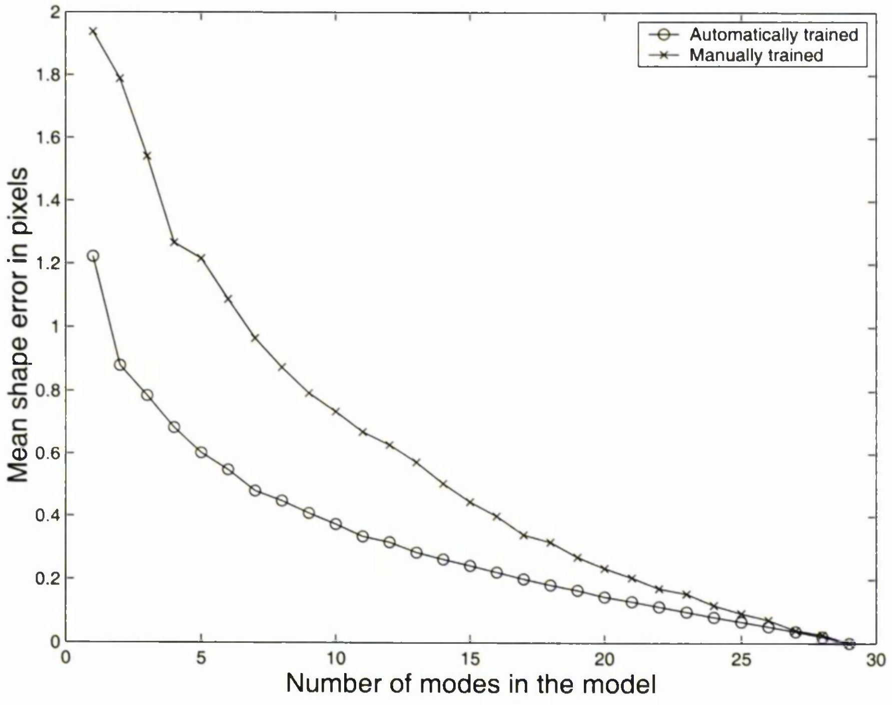


(b)

Figure 8.9: Compares the texture error (a) and shape error (b) of an automatically built model with that of an Eigenface model and a model trained from hand placed landmarks. The models were build using 30 frames of sequence B (Section 7.4).

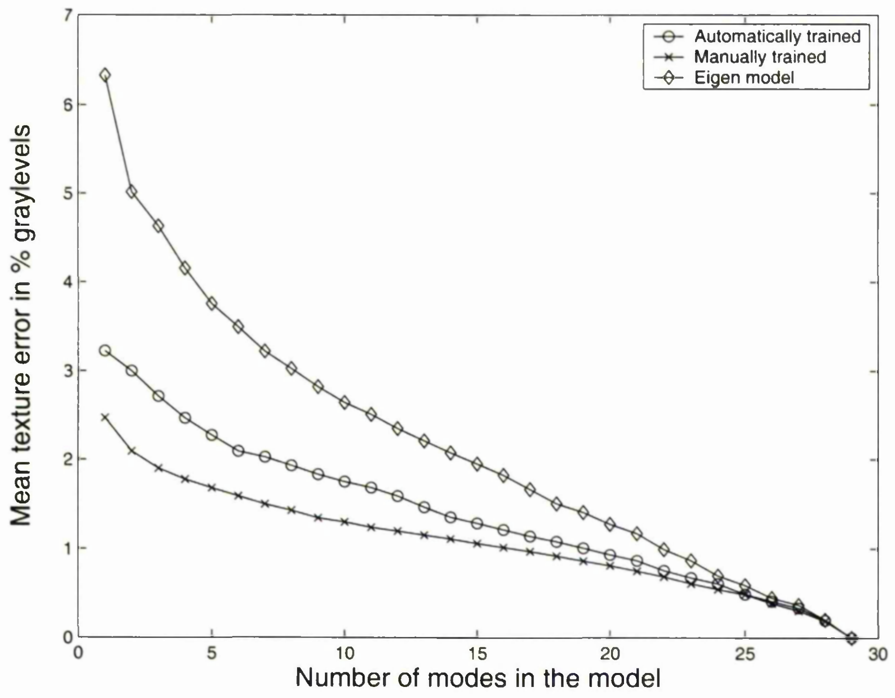


(a)

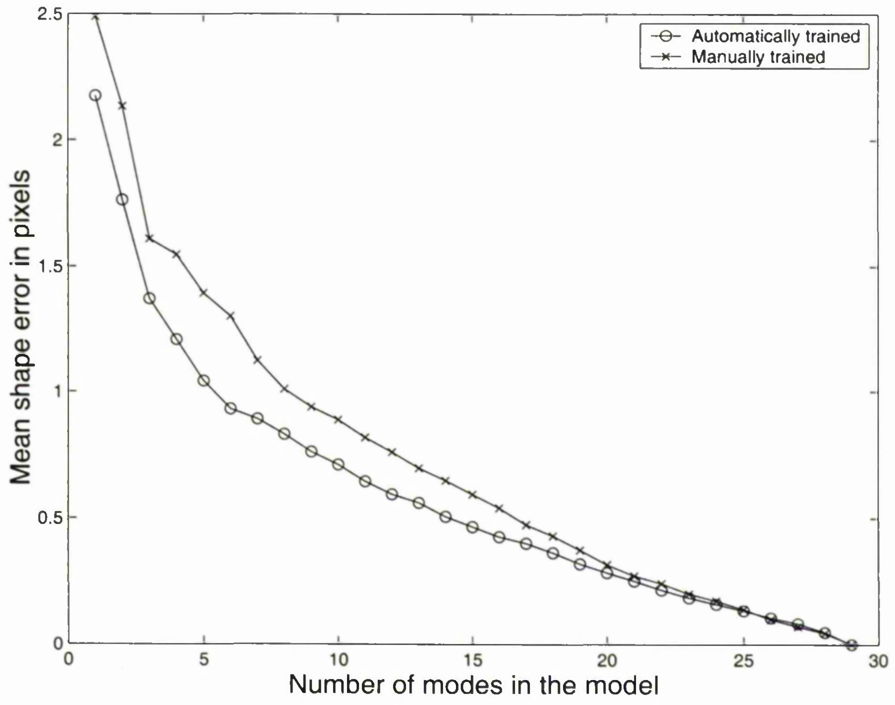


(b)

Figure 8.10: Compares the texture error (a) and shape error (b) of an automatically built model with that of an Eigenface model and a model trained from hand placed landmarks. The models were build using 30 frames of sequence C (Section 7.4).



(a)



(b)

Figure 8.11: Compares the texture error (a) and shape error (b) of an automatically built model with that of an Eigenface model and a model trained from hand placed landmarks. The models were build using 30 frames of sequence D (Section 7.4).

Chapter 9

A Multi-resolution framework for automatic model building

The previous chapter presented an iterative scheme in which an initial approximation to the true globally consistent transform is improved using pair-wise transforms. Although this approach was an improvement over the tracking scheme presented in Chaper 7, it still had difficulty in corresponding image sets in which the features moved relatively large distances, for instance in face image sets displaying a large amount of pose variation. In section 8.1.1 we made the assumption that the object's features will not move more than a given number, r, pixels between training examples. In image sets displaying a large amount of variation this doesn't hold true, and matching fails. One solution is to increase r for that image set so the assumption does hold, but this drastically increases the computation time as the number of comparisons required for each match is equal to πr^2 (see Section 8.1.1).

This chapter presents a multi-resolution framework which makes the system much more robust to extreme object deformations between training images without drastically increasing the computation time.

9.1 Multi-resolution framework

For each training image a Gaussian pyramid [12] is built (where level 0 is the original image and level L_{max} is the coarsest level). The scheme presented in the previous chapter is executed on the coarse level images to get an approximate globally consistent transform. At the coarse resolution we can search more of the image efficiently to obtain the correspondences between image pairs, allowing for larger deformations between images. This is because at a coarse resolution a search radius of r pixels covers a larger relative area than a search radius of r pixels at finer resolutions. This is illustrated in Figure 9.1. Each of the finer resolutions are processed in turn taking the globally consistent transform and refining it further.

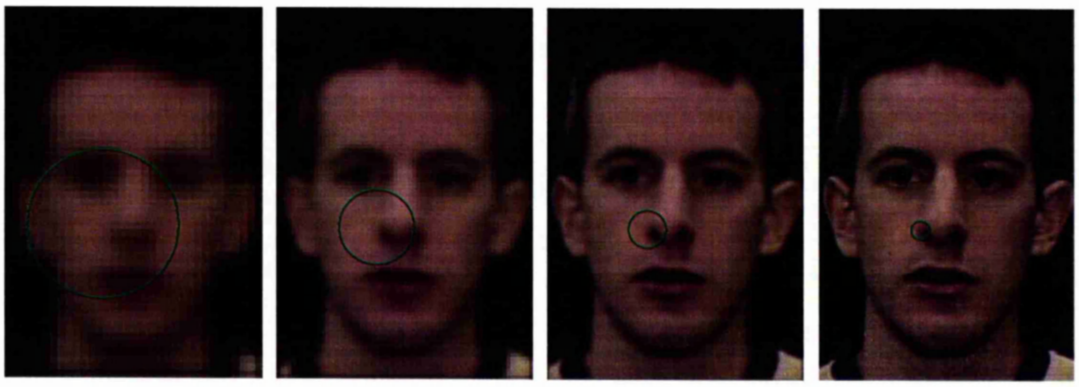


Figure 9.1: Illustrates how the size of the search area varies at different levels of a Gaussian pyramid for a constant search radius of $r = 10$. Note the time taken to search each of these regions is the same.

The following pseudo code further explains the multi-resolution framework.

```
1. Set $\boldsymbol{x}_i$ to the initialisation grid
2. Set $L = L_{max}$
3. While $L \geq 0$
   (a) For each image $i$
       i. Locate salient features, $\boldsymbol{s}_i$, at level $L$ from training image $i$
       ii. For each image $j \neq i$
           A. Predict the approximate positions, $\boldsymbol{m}'_{ij} = f(\boldsymbol{s}_i|\boldsymbol{x}_i, \boldsymbol{x}_j)$, of each of the salient features, $\boldsymbol{s}_i$, for image $i$ in image $j$
           B. Search around predicted matches, $\boldsymbol{m}'_{ij}$, for better matches, $\boldsymbol{m}_{ij}$, using equation 8.1
   (b) Define the set of pair-wise transformations, $\{T_{ji}\}$, using the salient features, $\boldsymbol{s}_i$, together with their matches, $\boldsymbol{m}_{ij}$, as shown by equation 8.8
   (c) Define a new globally consistent transformation, $\boldsymbol{x}_i$, using the iterative scheme defined in section 8.1.4
   (d) If $L > 0$ then $L \rightarrow (L - 1)$
4. The final result is the globally consistent transform as described by $\boldsymbol{x}_i$
```

Figure 9.2 illustrates how the grid, $\boldsymbol{x}_i$, deforms as each resolution is processed for 5 example training images.

If n_r is the number of resolutions searched, the computational complexity of the multi-resolution algorithm in terms of the amount of acceptable feature movement in pixels, r_p is $n_r(\frac{r_p}{2^{(n_r-1)}})^2$. To achieve the same amount of feature movement searching at a single resolution the computational complexity is ${r_p}^2$. So using a multi-resolution framework reduces the computational complexity by a factor of $\frac{4^{(n_r-1)}}{n_r}$.

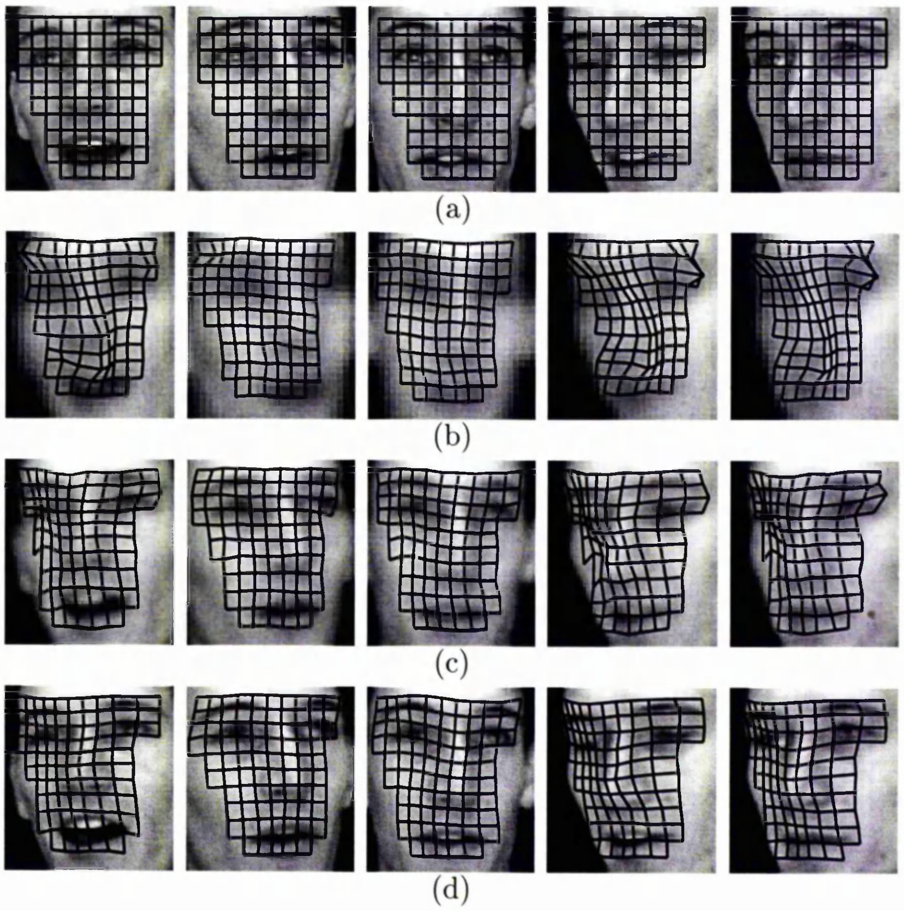


Figure 9.2: Example of how the grid deforms to represent the underlying shape during multi-resolution analysis. (a) shows how the grid is initialised, (b) after applying the algorithm at the coarsest resolution $L = L_{max}$, (c) after $L = L_{max} - 1$, (d) after convergence.

9.2 Results

We attempted to use the multi-resolution scheme to automatically landmark 4 training sets used in Section 7.3. The multi-resolution scheme used Gaussian pyramid levels one, two and three (see Figure 9.1). As a means of contrast we also repeated the experiment using the single resolution frameworks at Gaussian pyramid level one, two and three.

For each training set five different sets of landmarks exist, they are listed below:

- Hand placed landmarks
- Automatic landmarks generated from multi resolution framework using Gaussian pyramid levels one, two and three.
- Automatic landmarks generated from single resolution framework using training images from the first level of a Gaussian pyramid.
- Automatic landmarks generated from single resolution framework using training images from the second level of a Gaussian pyramid.
- Automatic landmarks generated from single resolution framework using training images from the third level of a Gaussian pyramid.

Using these landmarks, 5 models were built for each of the training sets. We assessed the five models in the same way as in the previous two chapters (see Sections 8.2 and 7.3). This involves measuring the shape and texture error incurred when reconstructing the training images with varying numbers of model modes.

The results showed that in all cases the multi resolution framework did at least as well as the best single resolution model. The most interesting result was with Sequence D (See Figure 7.4). Sequence D was the only training set in which the single resolution framework was unable to produce a model that was better than the one trained using manually placed landmarks (see Section 8.2). The multi resolution framework managed to produce an automatic model which has a smaller reconstruction error than the manually trained model, as shown in Figure 9.4. This is because the multi resolution framework was able to capture the large amount of pose variation displayed in sequence D.

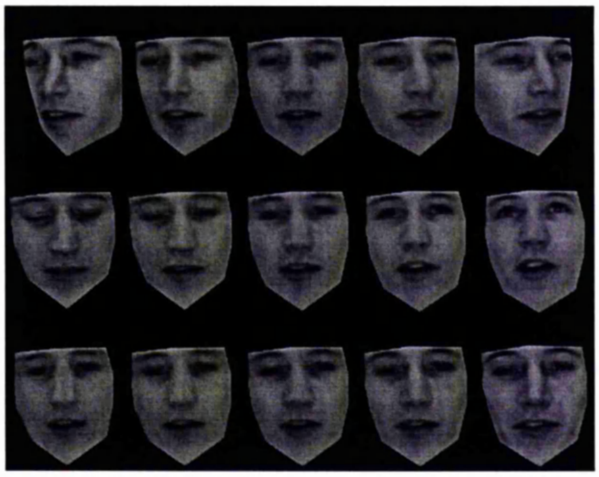
Figure 9.3(a) illustrates the principle modes of variation of the manually trained model, (c) shows the modes of the automatically trained model, and (b) shows the modes of the eigen model. Note that the modes of the manually trained and automatically trained model are of a similar quality when compared with those of the eigen model.

9.3 Discussion

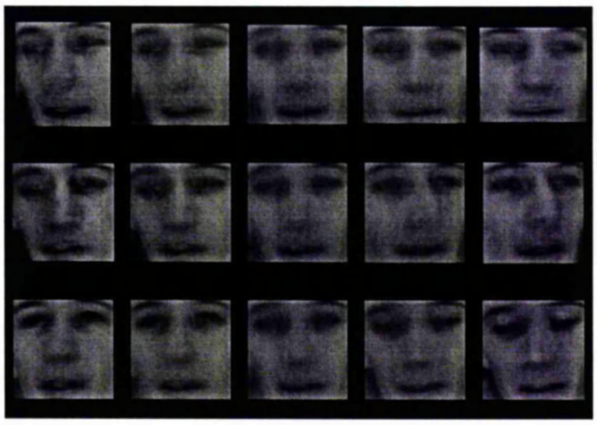
This Chapter presents a multi-resolution extension to the approach described in Chapter 8. Gaussian image pyramids are calculated for each of the training images. The multi-resolution scheme attempts to find a approximate globally consistent transform at the coarse resolution then uses the finer resolutions, in turn, to refine the transform.

Results have been presented comparing the multi-resolution framework to the single resolution method at multiple scales. For the training sets tested, the multi-resolution framework did at least as well as any given single resolution framework. The multi-resolution framework out-performed the single resolution framework on training set D. Training set D displays the most pose and expression variation of all the training sets tested. This indicates that the multi-resolution framework has improved the approach's robustness to the more extreme training set variations.

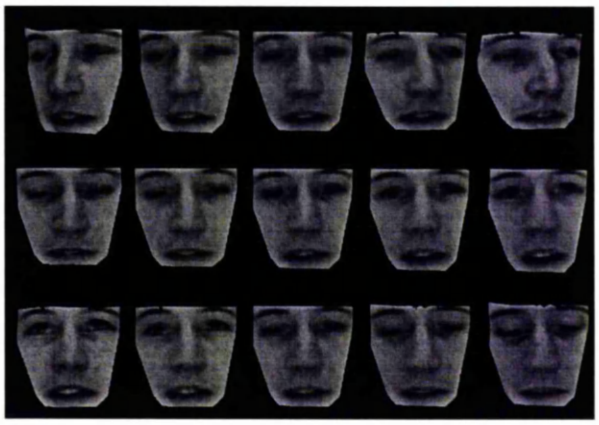
Some of the ideas presented in this chapter will be presented at the European Conference on Computer Vision 2000 [91].



(a)



(b)



(c)

Figure 9.3: The 3 most significant modes of variation for an manually trained (a), an eigen model (b) and automatically trained model (c). The modes in all cases are shown to ± 2.5 standard deviations.

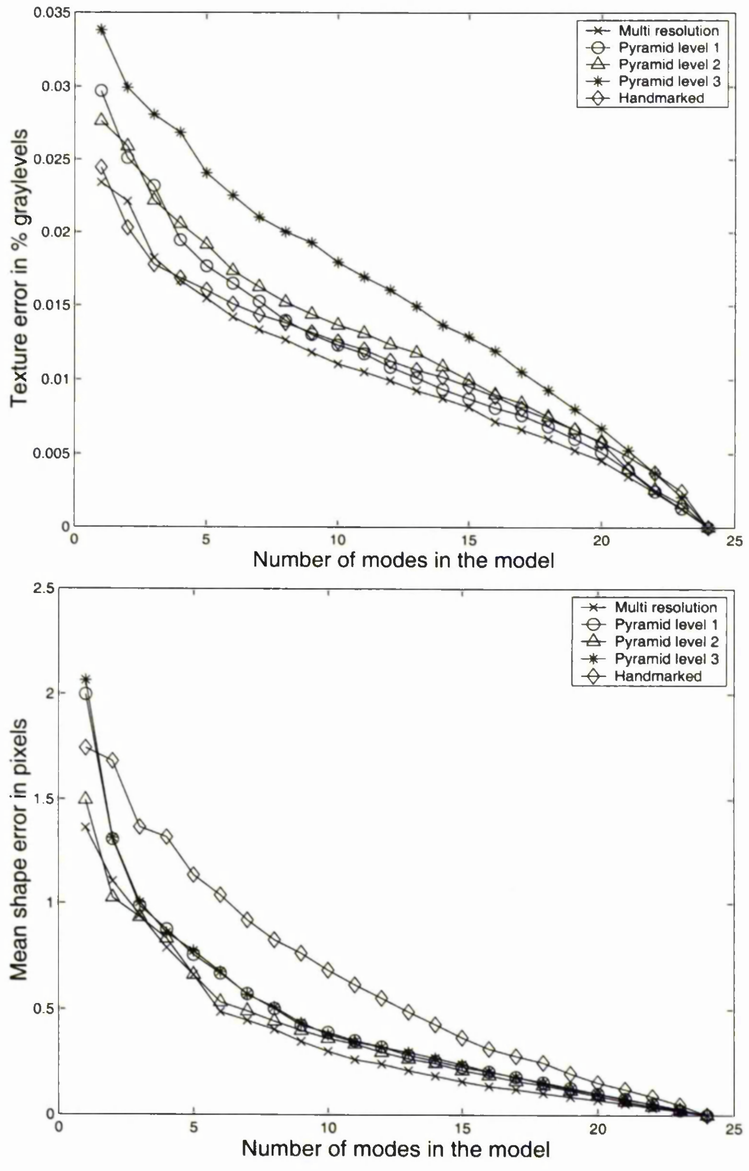


Figure 9.4: Compares the texture error (top) and shape error (bottom) of an automatically built model with that of a model trained from hand placed landmarks for sequence D.

Chapter 10

Conclusions

This thesis has described the development of automatic methods for training statistical models of shape and appearance. In this chapter we summarise the main achievements and conclusions of the research.

10.1 Measures of Feature Saliency

Chapters 5 and 6 developed two methods in which to measure a feature's saliency. They are known as *Image Feature Saliency* and *Object Feature Saliency*. We showed how measures of saliency can be used to select those object features which are less likely to generate false positive matches in subsequent examples of the object. Below we summarise the main differences between the two measures of saliency.

10.1.1 Image Feature Saliency

Image Feature Saliency is a measure of saliency based upon a single example image of the object class. Image Feature Saliency defines salient features as those which have a low probability of being misclassified with any other features within the single image containing the object. We presented quantitative results showing that salient features can be found with a greater degree of success in unseen object examples than either randomly selected features or manually selected features.

10.1.2 Object Feature Saliency

Object Feature Saliency differs from Image Feature Saliency in that it considers how features vary over a number of training examples for which there is a known correspondence. Object Feature Saliency defines salient features as those which have a low probability of being misclassified with any other features within the object class. We showed that features selected using Object Feature Saliency can be found with a greater degree of success in unseen object examples than features selected using Image Feature Saliency.

Both saliency measures chose features semi-independently and they are therefore not the optimum group. Finding the optimum group is an extremely complicated task, future research could address this problem. We applied the notion of saliency to improve the robustness of one object interpretation task, locating new instances of a human face. We also believe that other interpretation tasks can also benefit from the application of saliency.

10.2 Serial Approaches to Automatic Model Building

In Chapter 7 we developed a serial (frame by frame) approach to automatically building models from image sequences. Although the approach did manage to successfully build automatic models from 2 sequences, it failed of a further 2 sequences. Further improvements to the scheme could have been made, but we concluded that serial schemes to automatic model building are flawed. This is because serial schemes begin with a poorly training model of some sort, resulting in a high probability of computing a bad correspondence with first few training images. If bad correspondences are found early on in the sequence it will have the effect of corrupting the model, decreasing the chances of locating correct correspondences in subsequent frames (or training images). For this reason we turned are attention to developing parallel schemes.

A positive conclusion drawn from the chapter was that is possible to build appearance models automatically which are of a similar or better quality than appearance models built using manually corresponded training data. This highlighted the fact that models built using manually corresponded training data should not be seen as perfect models, they include human errors. Automatically built models can eliminate human errors.

10.3 Parallel Approaches to Automatic Model Building

Chapters 8 and 9 develop a parallel scheme for automatically building models from image sets. The system calculates globally inconsistent transformations between all image pairs. These transformations are used to drive an iterative scheme that calculates a globally consistent set of transforms. The globally consistent set of transforms are used to provide the correspondence necessary to build an appearance model. A multi-resolution framework is presented which improves the approach's robustness to extreme feature deformations. We used the parallel method to successfully build appearance models for all of 4 training sets. Moreover, the models were found to be more compact that the equivalent models built using a manually obtained correspondence. This result adds further weight to our claim that parallel approaches are the correct way to tackle automatic model building.

10.4 Future work

Results presented in this thesis have built models from a relatively small number of training examples, typically 25 to 50. This is because matching all pairs of training images is extremely time consuming. We would like to look at the effect each pair-wise match has on the overall global correspondence. If we could predict which combination of pairs of images have the most positive effect on the final global correspondence we could limit, in a sensible way, which image pairs to match. This would result in our approach being applicable to much larger training sets.

This thesis has been primarily concerned with building models of frontal views of the human face. The parallel method developed in Chapters 8 and 9 is applicable to any object class from which an appearance model can be built. We would like to explore our parallel method's suitability to these other object classes. Other object classes include the profile of the human face, 2D slices through the human brain and circuit board resistors.

The iterative scheme presented in Section 8.1.4 is a general method in which to obtain globally consistent transforms from pair-wise transforms. Throughout this thesis we have used feature vectors of Gaussian partial derivatives in order to compute pair-wise transforms. We would like to experiment with different schemes which compute the pair-wise transforms such as optical flow and Gabor wavelets. We would like to explore the possibility of automatically selecting the best pair-wise correspondence scheme for a set of image data.

10.5 Final statement

Automatic model building is a difficult but not insurmountable task. This thesis has presented research resulting in a feature based multi-resolution parallel scheme. The scheme is capable of building appearance models of the human face which are more compact than models trained using a correspondence obtained manually.

Appendix A

Warping Images

A.1 Interpolation

Suppose we have a pair of images with n correspondences defined between them. The correspondences are represented by two sets of n control points, $\boldsymbol{y}_i = \{\boldsymbol{y}_{ip}\}$ and $\boldsymbol{y}_j = \{\boldsymbol{y}_{jp}\}$ which describe the position of each correspondence in images $\boldsymbol{I}_i$ and $\boldsymbol{I}_j$ respectively.

Suppose $\boldsymbol{x}_i$ defines a second set of points in image $\boldsymbol{I}_i$ who's position in image $\boldsymbol{I}_j$ (defined as $\boldsymbol{x}_j$) is unknown. We would like to calculate the approximation position of $\boldsymbol{x}_j$ based on the set of control points $\boldsymbol{y}_i$ and $\boldsymbol{y}_j$.

Let $\boldsymbol{x}_j = \boldsymbol{f}_{ji}(\boldsymbol{x}_i|\boldsymbol{y}_i, \boldsymbol{y}_j)$ be a mapping from points $\boldsymbol{x}_i$ in image I_i to points $\boldsymbol{x}_j$ in image I_j, controlled by a set of control points $\boldsymbol{y}_i$ and $\boldsymbol{y}_j$.

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

$$\boldsymbol{f}_{ji}(\boldsymbol{x}|\boldsymbol{y}_i, \boldsymbol{y}_j) = \sum_{p=1}^{n} f_p(\boldsymbol{x})\boldsymbol{y}_{jp} \tag{A.1}$$

where $\boldsymbol{x}$ is an arbitrary point in image $\boldsymbol{I}_i$ and the n continuous scalar valued functions f_p each satisfy

$$f_p(\boldsymbol{y}_{iq}) = \begin{cases} 1 & if \quad p = q \\ 0 & \quad\; p \neq q \end{cases} \tag{A.2}$$

This ensures $\boldsymbol{y}_j = \boldsymbol{f}_{ji}(\boldsymbol{y}_i | \boldsymbol{y}_i, \boldsymbol{y}_j)$.

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

A.1.1 Piece-wise Affine

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

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

We would like to arrange that f_p will map a point x which is halfway between y_ip and $y_{i(p+1)}$ to a point halfway between y_jp and $y_{j(p+1)}$. This is achieved by setting

$$f_p(x) = \begin{cases} (x - y_{ip})/(y_{i(p+1)} - y_{ip}) & if \quad x \in [y_{ip}, y_{i(p+1)}] \; and \; p < n \\ (x - y_{ip})/(y_{ip} - y_{i(p-1)}) & if \quad x \in [y_{i(p-1)}, y_{ip}] \; and \; p > 1 \\ 0 & \quad\;\; otherwise \end{cases} \tag{A.3}$$

We can only sensibly interpolate values of x which lie in the region between the control points, $[y_{i1}, y_{in}]$.

In two dimensions, we can use a triangulation (eg Delauney) to partition the convex hull of the control points within each image 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 from image $\boldsymbol{I}_i$ to their new positions in image $\boldsymbol{I}_j$.

Suppose $\boldsymbol{y}_{i1}$, $\boldsymbol{y}_{i2}$ and $\boldsymbol{y}_{i3}$ are three corners of such a triangle. Any internal point can be written:

$$\begin{aligned} \boldsymbol{x} &= \boldsymbol{y}_{i1} + \beta(\boldsymbol{y}_{i2} - \boldsymbol{y}_1) + \gamma(\boldsymbol{y}_{i3} - \boldsymbol{y}_{i1}) \\ &= \alpha\boldsymbol{y}_{i1} + \beta\boldsymbol{y}_{i2} + \gamma\boldsymbol{y}_{i3} \end{aligned} \tag{A.4}$$

where $\alpha = 1 - (\beta + \gamma)$ and so $\alpha + \beta + \gamma = 1$. For $\boldsymbol{x}$ to be inside the triangle, $0 \leq \alpha, \beta, \gamma \leq 1$.

Under the affine transformation for a single point $\boldsymbol{x}$, this point simply maps to

$$\boldsymbol{x} = \boldsymbol{f}_{ji}(\boldsymbol{x}_i|\boldsymbol{y}_i, \boldsymbol{y}_j) = \alpha\boldsymbol{y}_{j1} + \beta\boldsymbol{y}_{j2} + \gamma\boldsymbol{y}_{j3} \tag{A.5}$$

where point $\boldsymbol{x}$ is a point in image $\boldsymbol{I}_i$ which lies within the triangle defined by control points $\boldsymbol{y}_{j1}$, $\boldsymbol{y}_{j2}$ and $\boldsymbol{y}_{j3}$.

To interpolate a set of points, $\boldsymbol{x}_i$ from image $\boldsymbol{I}_i$ we 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 image $\boldsymbol{I}_j$.

Note that although this gives a continuous deformation, it is not smooth.

A.1.2 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 piece-wise affine in that they are not constrained to the convex hull of the control points. However, they are more expensive to calculate.

A.1.3 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

$$f(x) = \sum_{p=1}^{n} w_{ip} U(|x - y_{ip}|) + a_0 + a_1 x \tag{A.6}$$

The weights w_{ip},a_0, a_1 are chosen to satisfy the constraints $f(y_{ip}) = y_{jp} \, \forall p$.

If we define the vector function

$$\boldsymbol{u}_1(x) = (U(|x - y_{i1}|), U(|x - y_{i2}|, \dots , U(|x - y_{in}|), 1, x)^T \tag{A.7}$$

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

then (A.6) becomes

$$f_1(x) = \boldsymbol{w}_1^T \boldsymbol{u}_1(x) \tag{A.8}$$

By plugging each pair of corresponding control points into (A.6) we get n linear equations of the form

$$y_{jq} = \sum_{p=1}^{n} w_{ip} U(|y_{iq} - y_{ip}|) + a_0 + a_1 y_{iq} \tag{A.9}$$

Let $U_{pq} = U_{qp} = U(|y_{iq} - y_{ip}|)$. Let $U_{pp} = 0$. Let $\boldsymbol{K}$ be the symetric $n \times n$ matrix whose elements are $\{U_{pq}\}$.

Let

$$\boldsymbol{Q}_1 = \begin{pmatrix} 1 & y_{i1} \\ 1 & y_{i2} \\ \vdots & \vdots \\ 1 & y_{in} \end{pmatrix} \tag{A.10}$$

$$\boldsymbol{L}_1 = \begin{pmatrix} \boldsymbol{K} & \boldsymbol{Q}_1 \\ \boldsymbol{Q}_1^T & \mathbf{0}_2 \end{pmatrix} \tag{A.11}$$

where $\mathbf{0}_2$ is a 2×2 zero matrix.

Let $\boldsymbol{X}_1 = (y_{j1}, y_{j2}, \ldots, y_{jn}, 0, 0)^T$. Then the weights for the spline (A.6) $\boldsymbol{w}_1 = (w_1, \ldots, w_n, a_0, a_1)$ are given by the solution to the linear equation

$$\boldsymbol{L}_1 \boldsymbol{w} = \boldsymbol{X}'_1 \tag{A.12}$$

A.1.4 Many Dimensions

The extension to the d-dimensional case is straightforward.

If we define the vector function

$$\boldsymbol{u}_d(\boldsymbol{x}) = (U(|\boldsymbol{x} - \boldsymbol{y}_{i1}|), \ldots, U(|\boldsymbol{x} - \boldsymbol{y}_{in}|), 1|\boldsymbol{x}^T)^T \tag{A.13}$$

then the d-dimensional thin plate spline is given by

$$\boldsymbol{f}(\boldsymbol{x}) = \boldsymbol{W}\boldsymbol{u}_d(\boldsymbol{x}) \tag{A.14}$$

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

To choose weights to satisfy the constraints, construct the matrices

$$\boldsymbol{Q}_d = \begin{pmatrix} 1 & \boldsymbol{y}_{i1}^T \\ 1 & \boldsymbol{y}_{i2}^T \\ \vdots & \vdots \\ 1 & \boldsymbol{y}_{in}^T \end{pmatrix} \tag{A.15}$$

$$\boldsymbol{L}_d = \begin{pmatrix} \boldsymbol{K} & \boldsymbol{Q}_d \\ \boldsymbol{Q}_d^T & \mathbf{0}_{d+1} \end{pmatrix} \tag{A.16}$$

where $\mathbf{0}_{d+1}$ is a $(d+1) \times (d+1)$ zero matrix, and $\boldsymbol{K}$ is a $n \times n$ matrix whose pq^{th} element is $U(|\boldsymbol{y}_{ip} - \boldsymbol{y}_{iq}|)$.

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

$$\boldsymbol{X}'_d = \begin{pmatrix} \boldsymbol{y}_{j1} \\ \vdots \\ \boldsymbol{y}_{jn} \\ \mathbf{0}_d \\ \vdots \mathbf{0}_d \end{pmatrix} \tag{A.17}$$

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

$$\boldsymbol{L}_d^T \boldsymbol{W}_d^T = \boldsymbol{X}'_d \tag{A.18}$$

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

Bibliography

[1] H. Asada and M. Brady. The curvature primal sketch. *IEEE Trans. Pattern Analysis and Machine Intelligence*, pages 2–14, 1986.

[2] F. Attneave. Some informational aspects of visual perception. *Psychological Review*, 61, 1954.

[3] D. R. Bailes and C. J. Taylor. The use of symmetry chords for expressing grey level constraints. In *British Machine Vision Conference*, pages 296–305. BMVC Press, 1995.

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

[5] D. H. Ballard and C. M. Brown. *Computer Vision.* Prentice-Hall, 1982.

[6] R. C. Bolles and R. A. Cain. Recognising and locating partially visible objects: the local-feature-focus method. *Int. J. Robotics Res*, 1:57–82, 1982.

[7] R. C. Bolles and R. A. Cain. 3dpo: A three-dimensional part orientation system. *Int. J. Robotics Res*, 5(3):3–26, 1986.

[8] 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.

[9] F. L. Bookstein. *Morphometric Tools for Landmark Data.* Cambridge University Press, 1991.

[10] A. C. Bovik, M. Clark, and W. S. Geisler. Multichannel texture analysis using localized spatial filters. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(12):55–73, 1990.

[11] A. D. Brett and C. J. Taylor. A framework for automated landmark generation for automated3D statistical model construction. In 16th *Conference on Information Processing in Medical Imaging*, Visegrád, Hungary, June 1999.

[12] P. Burt. *The Pyramid as a Structure for Efficient Computation*, pages 6–37. Springer-Verlag, 1984.

[13] B.W.Silverman. *Density Estimation*, page 87. Chapman and Hall, 1986.

[14] G. Christensen. Consistent linear-elastic transformations for image matching. In 16^{th} *Conference on Information Processing in Medical Imaging*, pages 224–237, Visegrád, Hungary, June 1999.

[15] 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*, pages 101–112. Kluwer Academic Publishers, 1995.

[16] T. Cootes, C. Beeston, G. J. Edwards, and C. Taylor. A unified framework for atlas matching using active appearance models. In 16^{th} *Conference on Information Processing in Medical Imaging*, page (To appear), 1999.

[17] T. 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, 1998.

[18] T. Cootes and C. Taylor. A mixture model for representing shape variation. *Image and Vision Computing*, page to Appear, 1999.

[19] T. Cootes, C. Taylor, and A. Lanitis. Active shape models : Evaluation of a multi-resolution method for improving image search. In E. Hancock, editor, 5^{th} *British Machine Vison Conference*, pages 327–336, York, England, Sept. 1994. BMVA Press.

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

[21] T. F. Cootes, A. Hill, and C. J. Taylor. Medical image interpretation using active shape models: Recent advances. In 14^{th} *Conference on Information Processing in Medical Imaging, France*, pages 371–372, June 1995.

[22] T. F. Cootes, A. Hill, C. J. Taylor, and J. Haslam. The use of active shape models for locating structures in medical images. In H. H. Barrett and A. F. Gmitro, editors, 13^{th} *Conference on Information Processing in Medical Imaging, Flagstaff, Arizona, USA*, pages 33–47. Springer-Verlag, June 1993.

[23] T. F. Cootes, A. Hill, C. J. Taylor, and J. Haslam. The use of active shape models for locating structures in medical images. *Image and Vision Computing*, 12(6):276–285, July 1994.

[24] T. F. Cootes and C. J. Taylor. Combining point distribution models with shape models based on finite-element analysis. *Image and Vision Computing*, 13(5):403–409, 1995.

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

[26] T. F. Cootes, C. J. Taylor, and A. Lanitis. Multi-resolution search with active shape models. In *Proc. International Conference on Pattern Recognition*, volume I, pages 610–612, Oct 1994.

[27] I. Craw and P. Cameron. Face recognition by computer. In 3^{rd} *British Machine Vision Conference*, pages 489–507, 1992.

[28] A. P. Dempster, N. M. Laird, and D. D. Rubin. Maximum likelihood from incomplete data via the em algorithm. *J.Royal Statist. Soc, Ser.B*, 39:1–38, 1977.

[29] R. Deriche and O. D. Faugeras. 2-d curve matching using high curvature points. In *10th International Conference on Pattern Recognition*, pages 240–242, June 1990.

[30] R. Deriche and G. Giraudon. A computational approach for corner and vertex detection. *International Journal of Computer Vision*, pages 101–124, 1993.

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

[32] G. Edwards, T. Cootes, and C. Taylor. Interpreting Face Images using Active Appearance Models. In 3^{rd} *International Conference on Automatic Face andGesture Recognition 1998*, pages 300–305, Nara, Japan, 1998.

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

[34] G. J. Edwards, A. Lanitis, C. J. Taylor, and T. Cootes. Statistical models of face images: Improving specificity. In 7^{th} *British Machine Vison Conference*, pages 765–774, Edinburgh, UK, 1996.

[35] G. J. Edwards, C. J. Taylor, and T. Cootes. Face recognition using the active appearance model. In H.Burkhardt and B. Neumann, editors, 5^{th} *European Conference on Computer Vision*, volume 2, pages 581–695. Springer, 1998.

[36] E. Elagin, J. Steffens, and H. Neven. Automatic pose estimation system for human faces based on bunch graph matching technology. In *International Conference on Automatic Face and Gesture Recognition*, 1998.

[37] F. Farrokhnia and R. Barber. A multi-channel filtering approach to texture segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 364–370, 1991.

[38] L. M. J. Florack, B. M. ter Haar Romeny, J. J. Koenderink, and M. A. Viergever. Scale and the differential structure of images. *Image and Vision Computing*, 10(6):376–388, July/August 1992.

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

[40] J. C. Gower. Generalized procrustes analysis. *Psychometrika*, 40:33–51, 1975.

[41] A. Hill and C. J. Taylor. Model-based image interpretation using genetic algorithms. *Image and Vision Computing*, 10(5):295–300, June 1992.

[42] A. Hill and C. J. Taylor. Automatic landmark generation for point distribution models. In E. Hancock, editor, 5th *British Machine Vison Conference*, pages 429–438. BMVA Press, Sept. 1994.

[43] 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.

[44] 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.

[45] H. Hong, H. Neven, and C. Malsburg. Online facial expression recognition based on personalized galleries. In *International Conference on Automatic Face and Gesture Recognition*, pages 354–359, 1998.

[46] R. Horaud, F. Veillon, and T. Skordas. Finding geometric and relational structures in images. In *1st European Conference on Computer Vision*, pages 374–384, 1990.

[47] R. A. Johnson and D. W. Wichern. *Applied Multivariate Statistical Analysis*. Prentice Hall, 1992.

[48] 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.

[49] P. Kalocsai, H. Neven, and J. Steffens. Statistical analysis of gabor-filter representation. In *International Conference on Automatic Face and Gesture Recognition*, pages 360–365, 1998.

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

[51] M. Kass, A. Witkin, and D. Terzopoulos. Snakes: Active contour models. In 1^{st} *International Conference on Computer Vision*, London, June 1987.

[52] D. C. Kay. *Tensor Calculus, Schaum's Outline.* McGraw-Hill, 1988.

[53] L. Kitchen and A. Rosenfeld. Gray level corner detection. *Pattern Recognition Letters*, 1:95–102, December 1982.

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

[55] 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.

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

[57] T. S. Lee. Image representation using 2d gabor wavelets. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 18(10):959–971, Oct 1996.

[58] H. Lester, S. A. Arridge, K. M. Jansons, L. Lemieux, J. V. Hajnal, and A. Oatridge. Non-linear registration with the variable viscosity fluid algorithm. In 16^{th} *Conference on Information Processing in Medical Imaging*, pages 238–251, Visegrád, Hungary, June 1999.

[59] M. Lyons and S. Akamatsu. Coding Facial Expressions with Gabor Wavelets. In 3^{rd} *International Conference on Automatic Face andGesture Recognition 1998*, pages 200–205, Nara, Japan, 1998.

[60] B. S. Manjunath and W. Y. Ma. Texture features for browsing and retrieval of image data. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 18(8):837–842, 1996.

[61] B. F. Manly. *Multivariate statistical methods: A Primer.* Chapman and Hall, London, 1986.

[62] G. McLachlan and K. Basford. Mixture models : Inference and applications to clustering. *Statistics : Textbooks and Monographs*, 84:37–70, 1988.

[63] P. M. Merlin and D. J. Farber. A parallel mechanism for detecting curves in pictures. *IEEE Trans. Comput.*, C-24:96–98, Jan 1975.

[64] T. N. Mudge, J. L. Turney, and R. A. Voltz. Automatic generation of salient features for the recognition of partially occluded parts. *Robotica*, 5:117–127, 1987.

[65] C. Naster and N. Ayache. Non-rigid motion analysis in medical images: A physically based approach. In *Proc. IPMI*, pages 17–32, 1993.

[66] 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.

[67] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery. *Numerical Recipes in C.* Cambridge University Press, 1992.

[68] J. M. S. Prewitt. Object enhancement and extraction. In B. S. Lipkin and A. Rosenfeld, editors, *Picture Processing and Psychopictorics.* New York: Academic Press, 1970.

[69] L. G. Roberts. Machine perception of three-dimensional solids. In J. P. T. et al, editor, *Optical and Electro-Optical Information.* Cambridge, MA : MIT Press, 1965.

[70] K. Rohr, M. Fornefett, and H. S. Stiehl. Approximating thin-plate splines for elastic registration: Integration of landmark errors and orientation attributes. In *Information Processing in Medical Imaging*, pages 252–265. Springer, June/July 1999.

[71] S. Romdhani, S. Gong, and A. Psarrou. A multi-view nonlinear active shape model using kernel PCA. In *Proc. 10th British Machine Vision Conference*, pages 483–492, 1999.

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

[73] S. Sclaroff and A. P. Pentland. Modal matching for correspondence and recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 17(6):545–561, 1995.

[74] P. P. Smyth, C. J. Taylor, and J. E. Adams. Automatic measurement of vertebral shape using active shape models. In 15th *Conference on Information Processing in Medical Imaging*, pages 441–446, 1997.

[75] S. Solloway, C. Hutchinson, J. Waterton, and C. Taylor. The use of active shape models for making thickness measurements of articular cartilage from MR images. *Magnetic Resonance in Medicine*, 37:943–952, 1997.

[76] 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.

[77] N. Sumpter, R. D. Boyle, and R. D. Tillett. Modelling collective animal behaviour using extended point distribution models. In A. F. Clark, editor, 8th *British Machine Vison Conference*, pages 242–251, 1997.

[78] B. M. ter Haar Romeny, L. M. J. Florack, A.H.Salden, and M. A. Viergiever. Higher order differential structure of images. *Info. Proc. in Medical Imaging*, pages 77–93, 1993.

[79] T.F.Cootes and C.J.Taylor. Active shape models - 'smart snakes'. In D. Hogg and R. Boyle, editors, 3rd *British Machine Vision Conference*, pages 266–275, September 1992.

[80] J. Triesch and C. von der Malsburg. Robust classification of hand postures against complex backgrounds. In *Automatic Face and Gesture Recognition*, pages 170–175, Los Alamitos, California, Oct. 1996. IEEE Computer Society Press.

[81] J. Triesh and C. von der Malsburg. Robust classification of hand postures against complex backgrounds. In *Automatic Face and Gesture Recognition*, pages 170–175, Los Alamitos, California, Oct. 1996. IEEE Computer Society Press.

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

[83] J. L. Turney, T. N. Mudge, and R. A. Voltz. Recognising partially occluded parts. *IEEE Trans. PAMI*, 7:410–421, 1985.

[84] T. Vetter, M. Jones, and T. Poggio. A bootstrapping algorithm for learning linearized models of object classes. In *Computer Vision and Pattern Recognition Conference*, pages 40–46, 1997.

[85] K. N. Walker, T. Cootes, , and C. J. Taylor. Correspondence based on distinct points using image invariants. In 8th *British Machine Vison Conference*, pages 540–549, Colchester, UK, 1997.

[86] K. N. Walker, T. Cootes, , and C. J. Taylor. Locating salient object features. In P. Lewis and M. Nixon, editors, 9th *British Machine Vison Conference*, volume 2, pages 557–566, Southampton, UK, Sept. 1998. BMVA Press.

[87] K. N. Walker, T. Cootes, , and C. J. Taylor. Automatically building appearence models from image sequences using salient features. In 10th *British Machine Vison Conference*, volume 2, pages 463–472, 1999.

[88] K. N. Walker, T. Cootes, , and C. J. Taylor. Locating salient facial features using image invariants. *Image and Vision Computing, Face Recognition Special Issue.*, To Appear.

[89] K. N. Walker, T. F. Cootes, and C. J. Taylor. Locating salient facial features using image invariants. In 3rd *International Conference on Automatic Face andGesture Recognition 1998*, pages 242–247, Nara, Japan, 1998.

[90] K. N. Walker, T. F. Cootes, and C. J. Taylor. Determining correspondeences for statistical models of facial appearance. In *4th International Conference on Automatic Face and Gesture Recognition 2000*, pages 272–276, 2000.

[91] K. N. Walker, T. F. Cootes, and C. J. Taylor. Determining correspondences for statistical models of appearance. In *European Conference on Computer Vision*, 2000.

[92] H. Wang and J. M. Brady. Corner detection: Some new results. *IEE Colloquium Digest of Systems Aspects of Machine Perception and Vision*, pages 1.1–1.4, 1992.

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

[94] L. Wiskott, J. Fellous, N.Kruger, and C. von der Malsburg. Face recognition and gender determination. In M. Bichsel, editor, *Automatic Face and Gesture Recognition*, pages 92–97, Switzerland, June 1995. MultiMedia lab. University of Zurich.