Locating Facial Features with Active Shape Models

Stephen Milborrow

Supervised by Dr Fred Nicolls

Submitted to the Faculty of Engineering, University of Cape Town,
in partial fulfillment of the requirements for the
Degree of Master of Science

Revision History: 1 Initial 2,3,4 Minor fixes

November 16, 2007

Abstract

This dissertation focuses on the problem of locating features in frontal views of upright human faces. The dissertation starts with the Active Shape Model of Cootes et al. [20] and extends it with the following techniques:

(1) selectively using two- instead of one-dimensional landmark profiles
(2) stacking two Active Shape Models in series
(3) extending the set of landmarks
(4) trimming covariance matrices by setting most entries to zero
(5) using other modifications such as adding noise to the training set.

The resulting feature locater is shown to compare favorably with previously published methods.

Declaration

I, Stephen Milborrow, declare that this dissertation is my own work except where otherwise stated. It is being submitted for the degree of Master of Science in Engineering at the University of Cape Town and it has not been submitted before for any degree or examination at any other university.

Signature of Author .....

August 2007 Petaluma CA

To my parents, Geoff and Libby.

Contents

Acknowledgments

I would like to thank Gerhard de Jager and Fred Nicolls for providing the best possible environment for me to do this work. I would also like to thank David Cristinacce for his gentlemanly support. A thanks goes out to the people who prepared and published the databases used in this project, and to those who allowed their faces to be used.

Chapter 1

Introduction

This dissertation evaluates some variations of the Active Shape Model of Cootes et al. [20], as applied to monochrome front views of upright faces with neutral expressions. We will see that although Active Shape Models are well studied, there is still room for improvement.

The reader is assumed to be familiar with basic image processing and linear algebra. Familiarity with Active Shape Models is not assumed; these will be described from the ground up.

1.1 The problem

The problem under investigation is this: given an image of a face, find the position of a number of landmarks on the face. A *landmark* represents a distinguishable point present in most of the images under consideration. An example landmark is the location of the left eye

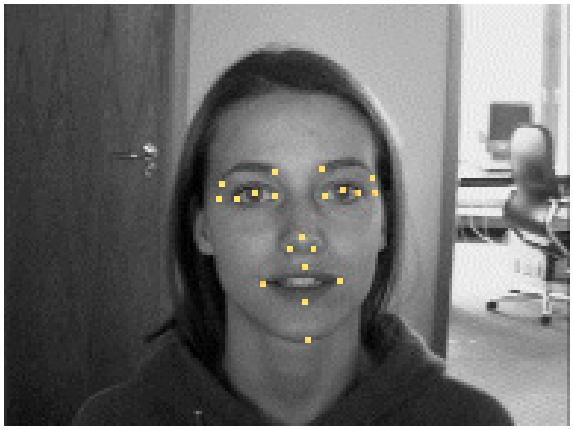


Figure 1.1: *A face with correctly positioned landmarks.*
All printed faces in this document are from the BioId set [44], except where noted.

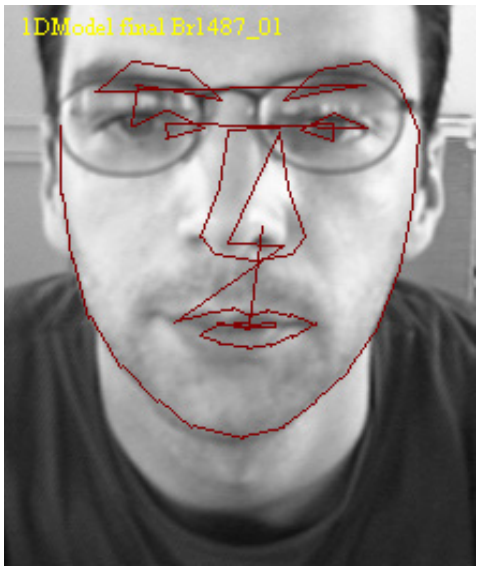


Figure 1.2: *Results of an unsatisfactory ASM search.*
Note the misplacement of nose and other landmarks.

pupil. Figure 1.1 on the previous page shows a face with landmarks in their correct positions.

Locating landmarks on faces is equivalent to locating *facial features*, because landmarks mark out facial features.

This project investigates solutions to the problem using the Active Shape Model (ASM) of Cootes et al. [20]. Some variations of the ASM will be presented and evaluated. We look only at variations that stay within the "iterate the profile model and shape model" strategy of the ASM, with separate profile and shape models. (Chapter 2 explains what this means.) An example variation under the scope of the project is using a two- instead of a one-dimensional search template around a landmark.

We look for improvements to the ASM because it is often inaccurate and sometimes fails completely. Figure 1.2 above shows the results of a search that is only partially successful.

1.2 Not the problem

Here are several related problems and techniques that are not part of this project.

1. Locating landmarks in features in **images other than faces**. Only faces are considered.
2. Finding the **overall position of face**. The ASM does a *local* search for facial features. The overall position of the face itself must first be approximately located before the ASM kicks in. Figure 1.3 on the next page shows the results of a face detector. This project for the most part uses the Rowley face and eye detector [58], but the Viola Jones detector [67] is also considered.

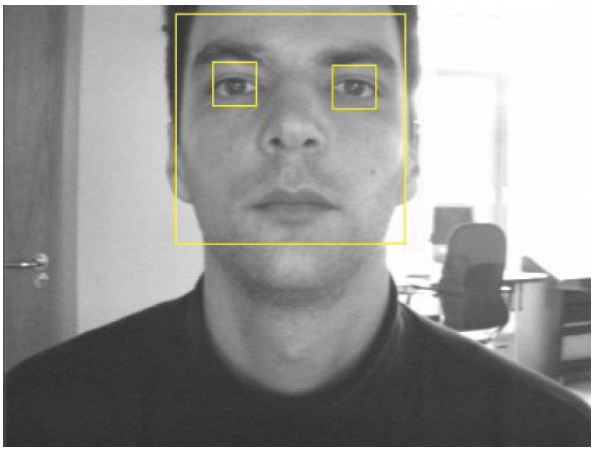


Figure 1.3: *An image with face and eyes located.*
The face and eyes were located by the Rowley global face detector and the ASM search can now begin.

3. Dealing with **all varieties of faces**. This project limits itself to front views of upright faces with neutral expressions.
4. **Recognizing the face** i.e. "saying who it is."
5. Recognizing **the facial expression** e.g. smiling, frowning. There are many schemes (a few derived from the ASM) to deal with this and the previous problem. We are concerned only with locating landmarks.
6. Using **3D image information**. We restrict ourselves to a 2D approach.
7. Using **color** during search. We use gray images.
8. Consolidating **multiple images** using motion clues, etc. We search for landmarks using a single image.
9. Using **Active Appearance Models (AAMs)**. The AAM is derived from the ASM and is described in Cootes et al. [13]. This project sticks to the ASM.

1.3 Layout of this document

This document is laid out as follows. Chapter 2 describes ASMs. Chapter 3 gives a brief history and reviews the literature. Chapter 4 looks at general aspects of model building and evaluation, and includes descriptions of the face databases used in this project. Chapter 5 works through some variations and parameters of the ASM. Chapter 6 summarizes results and compares them to published results. Chapter 7 discusses possible future research and wraps things up.

Chapter 2

Active Shape Models

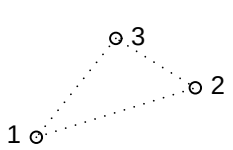
This chapter first describes shape models in general. It then gives a description of the classical Active Shape Model (ASM).

The *classical* ASM (my terminology) is that presented in Cootes and Taylor [20] Chapter 7. In brief, the classical ASM is characterized by its use of the Mahalanobis distance on one-dimensional profiles at each landmark and a linear point distribution model. Training determines the characteristics of the profile and point distribution models. What this all means will be explained shortly.

The account skips over aspects of the ASM that are not re-examined in this project. Cootes and Taylor [20] is the best comprehensive description of ASMs. Please refer to it where further details are needed.

2.1 Shapes and shape models

This section describes two-dimensional shapes and shape models in general. This theory forms the foundation on which the ASM is built.



	x	y
Point1	0.0	0.0
Point2	3.0	1.0
Point3	2.0	2.0

0.0
3.0
2.0
0.0
1.0
2.0

Figure 2.1: ***Left*** *a simple shape, with three points.*
Middle *the same shape as an array.*
Right *the shape as a vector.*

input set of unaligned shapes

1. Choose a reference shape (usually the first shape)
2. Translate each shape so that it is centered on the origin
3. Scale the reference shape to unit size. Call this shape $\bar{\boldsymbol{x}}_0$, the initial mean shape.
4. **repeat**
5. Align all shapes to the mean shape
6. Recalculate the mean shape from the aligned shapes
7. Constrain the current mean shape (align to $\bar{\boldsymbol{x}}_0$, scale to unit size)
8. **until** convergence (i.e. mean shape does not change much)

output set of aligned shapes, and mean shape

Figure 2.2: *Aligning shapes.*

2.1.1 Shapes

For our purposes, a *shape* is just an $n \times 2$ ordered set of points i.e. an array of (x,y) coordinates (Figure 2.1 on the previous page). The points are related to each other in some invariant sense. If you move a shape, it is still the same shape. If you expand or rotate it, it is still the same shape. *Edges* between points are not part of the shape but are often drawn to clarify the relationship or ordering between the points.

In practice it is convenient to represent a shape not as an $n \times 2$ array of (x,y) coordinates, but as a $2n \times 1$ vector: first all the x- and then all the y-coordinates. This representation is used for the equations in this report.

The *distance between two points* is the euclidean distance between the points. The *distance between two shapes* is the sum of the distances between their corresponding points. The *Procrustes distance* between two shapes $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$ is the root mean square distance between the shape points $\sqrt{(\boldsymbol{x}_1 - \boldsymbol{x}_2).(\boldsymbol{x}_1 - \boldsymbol{x}_2)}$ after alignment (alignment will be described shortly).

The *centroid* $\bar{\boldsymbol{x}}$ (also simply called the *position*) of a shape $\boldsymbol{x}$ is the mean of the point positions. The *size* of a shape is the root mean square distance between the shape points and the centroid.

There are other measures of distance and size but they are not used in this document.

2.1.2 Aligning shapes

A shape can be *aligned* to another shape by applying a transform which yields the minimum distance between the shapes. For our purposes, allowed transforms are scaling, rotating, and linear translation. A transform that does all three (but nothing else, such as shearing) is a *similarity* transform. The similarity transform $\boldsymbol{T}$ which rotates the point (x,y) by θ, scales it

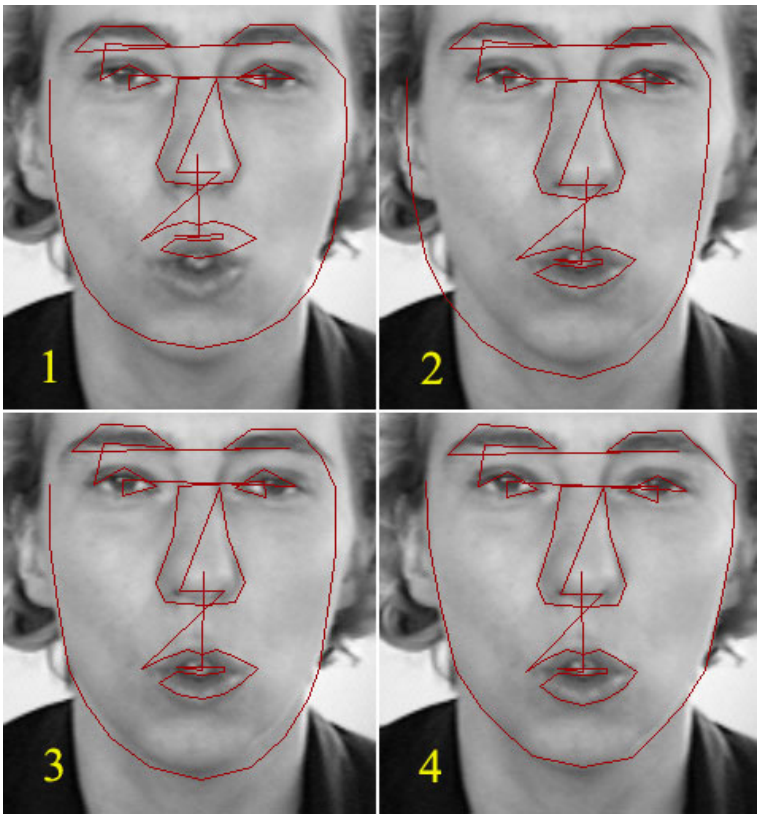


Figure 2.3: *An ASM search.*
Each picture shows the shape after correction by the shape model. Not all intermediate shapes are shown.

by s, and translates it by $x_{translate}$ and $y_{translate}$ is

$$\boldsymbol{T}\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} x_{translate} \\ y_{translate} \end{pmatrix} + \begin{pmatrix} s\ cos\ \theta & s\ sin\ \theta \\ -s\ sin\ \theta & s\ cos\ \theta \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} . \tag{2.1}$$

Cootes and Taylor [20] Appendix B gives methods to align two shapes by using a least-squares procedure. Accurate alignment may be deemed more important for certain points than for others, and points can be weighted accordingly during alignment.

A set of shapes can be aligned using an iterative algorithm reminiscent of k-means clustering, as shown in Figure 2.2 on the previous page. The constraint in line 7 is necessary to prevent the estimated mean shape from wandering around or shrinking.

Scaling and rotating shapes during alignment introduce non-linearities. These can be be minimized by projecting shapes into a tangent space ([20] section 4.1), but tangent spaces are not used in this project.

input image of a face

1. Generate the start shape by locating the overall position of the face
2. **repeat**
3. Suggest a new shape by profile matching around each shape point
4. Adjust the suggested shape to conform to the Shape Model
5. **until** convergence (i.e. until no further improvements in fit are possible)

output shape giving the (x,y) coordinates of the face landmarks

Figure 2.4: *ASM search algorithm for faces.*

2.1.3 Shape models

A *shape model* defines an allowable set of shapes. In this document, shape models have a fixed number of points and a matrix formula which specifies the relationship between the points. We will use *Point Distribution Models*, which learn allowable constellations of shape points from training examples and use principal components to build the model. A concrete example will be presented shortly.

2.2 Overview of the Active Shape Model

This section gives an overview of Active Shape Models. The following two sections then describe the two sub-models of the ASM: the shape model and the profile model. The phrase "active shape model" is sometimes used in a more general sense, but not in this document.

The ASM is first trained on a set of manually landmarked images. By *manually landmarked* we mean that somebody had to mark all the images by hand. This is done before training begins.

After training we can use the ASM to search for features on a face. An example search is shown in Figure 2.3 on the previous page. The general idea is (1) try to locate each landmark independently, then (2) correct the locations if necessary by looking at how the landmarks are located with respect to each other. To do this, the ASM is constructed from two types of sub-model:

1. a *profile model* for each landmark, which describes the characteristics of the image around the landmark. The model specifies what the image is expected to "look like" around the landmark. During training, we sample the area around each landmark across all training images to build a profile model for the landmark. During search, we sample the area in the vicinity of each tentative landmark, and move the landmark to the position that best matches that landmark's model profile. This generates tentative new positions for the landmarks, called the *suggested shape*.

input image of a face

1. Generate the start shape by locating the overall position of the face
2. **repeat**
3a. **for** each shape point
3b. **for** each offset (i.e. look in the area around the point)
3c. Build a profile by sampling the area around the offset
3d. Measure the fit of the profile against the Profile Model
3e. Move the point to the offset of the best profile match
4. Adjust the suggested shape to conform to the Shape Model
5. **until** convergence (i.e. until no further improvements in fit are possible)

output shape giving the (x,y) coordinates of the face landmarks

Figure 2.5: *ASM search algorithm for faces, with expanded profile matching step.*

2. a *shape model* which defines the allowable relative position of the landmarks. During search, the shape model adjusts the shape suggested by the profile model to conform to a legitimate face shape. This is needed because the profile matches at each landmark are unreliable. For example, the shape model will shrink the nose if the suggested face has a nose like Pinocchio.

The algorithm iterates for a solution using both sub-models as shown in Figure 2.4 on the previous page and in more detail in Figure 2.5 above.

The algorithm combines the results of the weak profile classifiers to build a stronger overall classifier. It is a shape constrained feature detector: the shape model acts globally; each profile matcher acts locally.

2.3 The ASM Shape Model

The job of the shape model is to convert the shape suggested by the profile models to an allowable face shape. Before building the shape model, the training shapes are aligned. Then the shape model consists of an average face and allowed distortions of the average face:

$$\hat{\boldsymbol{x}} = \bar{\boldsymbol{x}} + \boldsymbol{\Phi b} \tag{2.2}$$

where

$\hat{\boldsymbol{x}}$ is the generated shape vector (all the x- followed by all the y-coordinates). The hat on $\hat{\boldsymbol{x}}$ reminds us that it is generated by a model.

$\bar{\boldsymbol{x}}$ is the mean shape: the average of the aligned training shapes $\boldsymbol{x}_i$, defined as

$$\bar{\boldsymbol{x}} = \frac{1}{n_{\text{shapes}}} \sum_{i=1}^{n_{\text{shapes}}} \boldsymbol{x}_i \ . \tag{2.3}$$

Figure 2.6: *The mean face (black) with variations of the first principal component (gray).*

$\boldsymbol{\Phi}$ is the matrix of eigenvectors of the covariance matrix $\boldsymbol{S}_s$ of the training shape points

$$\boldsymbol{S}_s = \frac{1}{n_{\text{shapes}} - 1} \sum_{i=1}^{n_{\text{shapes}}} (\boldsymbol{x}_i - \bar{\boldsymbol{x}})(\boldsymbol{x}_i - \bar{\boldsymbol{x}})^T \ . \tag{2.4}$$

Using a standard principal components approach [45, 62], we order the eigenvalues λ_i of $\mathbf{S}_s$, and keep a limited number of the corresponding eigenvectors in $\boldsymbol{\Phi}$. The retained columns of $\boldsymbol{\Phi}$ are the eigenvectors corresponding to the largest eigenvalues of $\mathbf{S}_s$.

In this project, empirical testing determines the optimum number of eigenvectors to retain. We want to capture important aspects of the training shapes but ignore noise.

2.3.1 Generating shapes from the shape model

We can use Equation 2.2 to generate different shapes by varying the vector parameter $\boldsymbol{b}$. By keeping the elements of $\boldsymbol{b}$ within limits we ensure that generated faces are lifelike.

Figure 2.6 shows three faces generated by setting b_1 to $-3\sqrt{\lambda_1}$, 0, and $+3\sqrt{\lambda_1}$, with all other b_i's fixed at 0 (where b_1 is the first element of $\boldsymbol{b}$ and λ_1 is the largest eigenvalue).

2.3.2 Understanding the shape model

It is easier to gain an intuitive understanding of the shape model if we use rectangles instead of complicated face shapes. Figure 2.7 on the next page shows an example. To specify the four points of any of these rectangles, we need eight numbers: four x- and four y-coordinates. But the rectangles are centered and symmetrical about the origin. Thus we can actually specify a rectangle by just two parameters: its width and its height.

Now we use Equations 2.3 and 2.4 to generate the shape model $\hat{\boldsymbol{x}} = \bar{\boldsymbol{x}} + \boldsymbol{\Phi}\boldsymbol{b}$ from the given

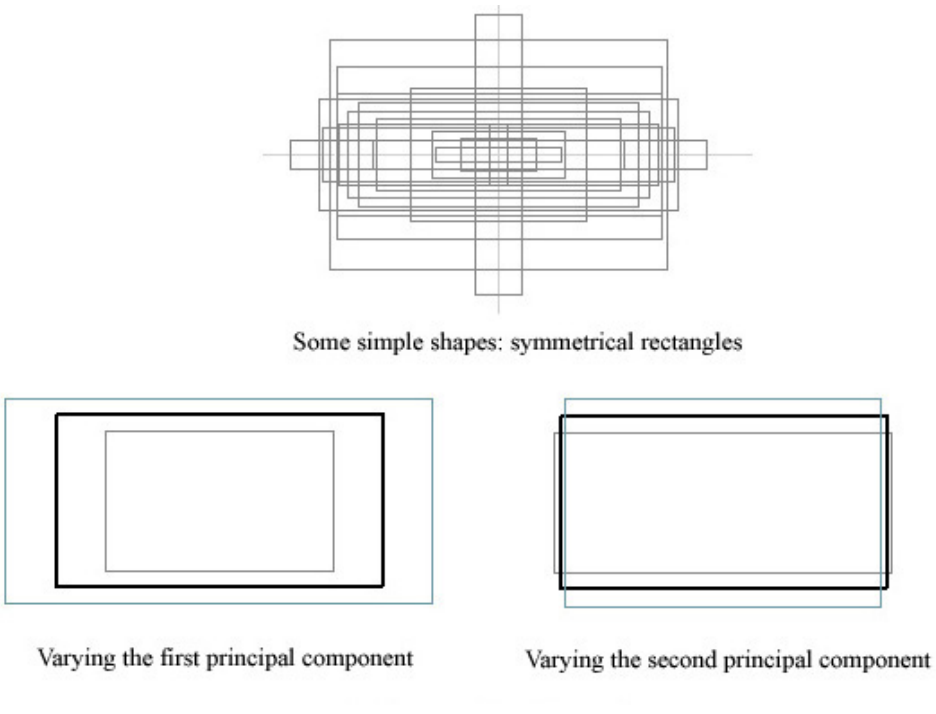


Figure 2.7: *A simple shape model.*
Top *a set of rectangular shapes, symmetrical about the origin.*
Left *the mean shape and two variations created by adding a multiple of the* ***first*** *eigenvector.*
Right *two variations created by adding a multiple of the* ***second*** *eigenvector.*

rectangles, which becomes[1]

$$\hat{\boldsymbol{x}} = \begin{pmatrix} 23 & 12 \\ -23 & 12 \\ -23 & -12 \\ 23 & -12 \end{pmatrix} + b_0 \begin{pmatrix} 12 & 4 \\ -12 & 4 \\ -12 & -4 \\ 12 & -4 \end{pmatrix} + b_1 \begin{pmatrix} -4 & 12 \\ 4 & 12 \\ 4 & -12 \\ -4 & -12 \end{pmatrix} + \dots \ .$$

The sorted eigenvalues of the covariance matrix $\boldsymbol{S}_s$ are $3778, 444, 2, 0.1, \dots$. There are eight eigenvalues altogether. The first two are much larger than the rest. (The remaining eigenvalues represent noise in the form of numerical errors.) Thus the process has discovered that the shapes can be parameterized by just two parameters, b_0 and b_1. The first parameter varies the contribution of the first eigenvector and, in this example, chiefly changes the size of the generated shape. The second parameter varies the contribution of the second eigenvector, which mainly adjusts the aspect ratio of the shape.

What is the advantage of this shape model over the "obvious" model where the two parameters simply specify the width and height? The advantage is that the first eigenvector captures as much variation as possible. If you had to specify a shape using just a single parameter, you

[1]The 8x1 vectors are reshaped into 4x2 matrices for clarity. The left column of each matrix shows x values; the right column shows y values.

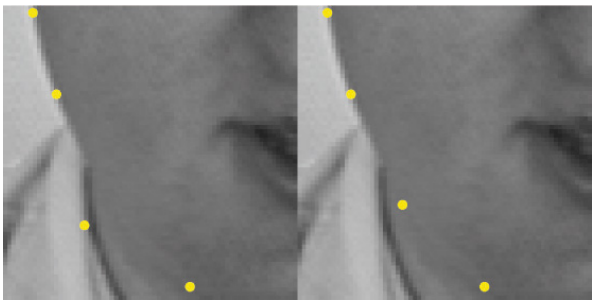


Figure 2.8: *Operation of the shape model.*
Left *landmarks located by the profile matcher.*
Right *the same landmarks conformed to the shape model. The landmark that was stuck on the collar is now better positioned.*

would on the whole get closer to your desired shape by adding a multiple of the first eigenvector to the mean shape, rather than by just changing the width. The second eigenvector captures as much of the remaining variation as possible, with the constraint that the eigenvectors are orthogonal.

For face shapes there will be many eigenvalues and no abrupt cutoff point. The relative size of the eigenvalues tells us the proportion of variation captured by the corresponding eigenvectors. We can capture as much variation of the input shapes as we want by retaining the appropriate number of eigenvectors.

2.3.3 Representing a given shape by the shape model

In the reverse direction, given a suggested shape $\boldsymbol{x}$ on the image, we can calculate the parameter $\boldsymbol{b}$ that allows Equation 2.2 to best approximate $\boldsymbol{x}$ with a model shape $\hat{\boldsymbol{x}}$. We seek the $\boldsymbol{b}$ and $\boldsymbol{T}$ that minimizes

$$distance(\boldsymbol{x},\ \boldsymbol{T}(\bar{\boldsymbol{x}} + \boldsymbol{\Phi b})) \ . \tag{2.5}$$

$\boldsymbol{T}$ is a similarity transform which maps the model space into the image space. The transform is needed because the face shape $\boldsymbol{x}$ could be anywhere in the image plane, but the model works off scaled upright shapes positioned on the origin. Cootes and Taylor [20] section 4.8 describes an iterative algorithm for finding $\boldsymbol{b}$ and $\boldsymbol{T}$.

After calculating $\boldsymbol{b}$, we reduce any out-of-range elements b_i to ensure that the generated $\hat{\boldsymbol{x}}$ conforms to the model, yet remains close to the suggested shape. We defer details to section 5.4.3.

2.3.4 An example

Now for an example of the Shape Model at work. The left side of Figure 2.8 above shows four landmarks, one of which is mispositioned by the profile matcher — it has snagged on

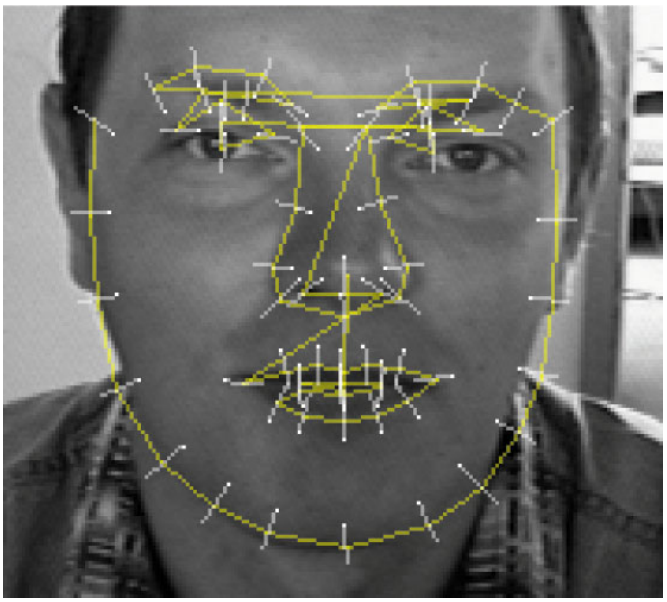


Figure 2.9: *The "mean face" (yellow) positioned over a search face at the start of a search. The white lines are "whiskers", along which the image intensity will be sampled to form profiles. In this example it so happens that the nose and mouth are already positioned quite accurately by the global face detector. The eyes are badly positioned.*

the collar. The right side shows the landmarks after correction to the shape model. As a side effect of the correction, a misplaced landmark may pull other landmarks slightly away from their (possibly correct) positions. This effect is not visible in the figure.

2.4 The ASM Profile Model

The job of the profile model is to take an approximate face shape and produce a better suggested shape by template matching at the landmarks. We start off the search with the mean face from the shape model, aligned and positioned with a global face detector (Figure 2.9).

This section describes the one-dimensional (1D) profiles used in the classical ASM. Later we will look at 2D profiles (section 5.5).

2.4.1 Forming a profile

To form the profile vector $\boldsymbol{g}$ at a landmark, we sample image intensities along a one-dimensional whisker. The *whisker* is a vector at the landmark which is orthogonal to a shape edge (Figure 2.10 on the next page). The profile vector is formed as follows:

1. Set each element of the profile vector to the gray level (0...255) of the image below it.

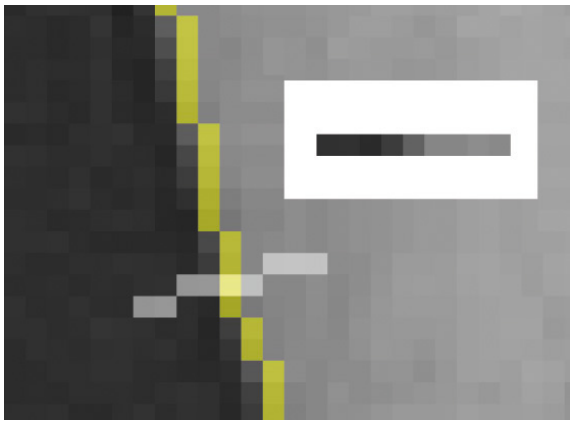


Figure 2.10: *A one-dimensional profile.*
The yellow line is the shape boundary. The gray line is the whisker, orthogonal to the boundary. The landmark is at the intersection of the shape boundary and the whisker. The inset shows the image intensity along the whisker. In practice, the ASM uses the normalized gradient of the image intensity.

2. Replace each profile element by the intensity gradient. This is done by replacing the profile element at each position i with the difference between it and the element at $i-1$.

3. Divide each element of the resulting vector by the sum of the absolute values of all vector elements.

Using normalized gradients in this manner is intended to lessen the effect of varying image lighting and contrast.

2.4.2 Building the profile model during training

During training, we build a model for each landmark by creating a mean profile $\bar{\boldsymbol{g}}$ and a covariance matrix $\boldsymbol{S}_g$ of all training profiles (one from each image) at that landmark. The assumption is that the profiles are approximately distributed as a multivariate Gaussian, and thus can be described by their mean and covariance matrix.

If the length of the profile is 7 (as in Figures 2.10 and 2.11), $\bar{\boldsymbol{g}}$ will have 7 elements. and $\boldsymbol{S}_g$ will be a 7×7 matrix. If there are 68 landmarks then there will be 68 separate $\bar{\boldsymbol{g}}$'s and $\boldsymbol{S}_g$'s.

2.4.3 Searching for the best profile

During search, at each landmark we form several search profiles by sampling the image in the neighborhood of the landmark. Each search profile is centered at small positive or negative displacements along the whisker. We typically form profiles at offsets up to about ± 3 pixels along the whisker. Figure 2.11 on the next page shows this process.

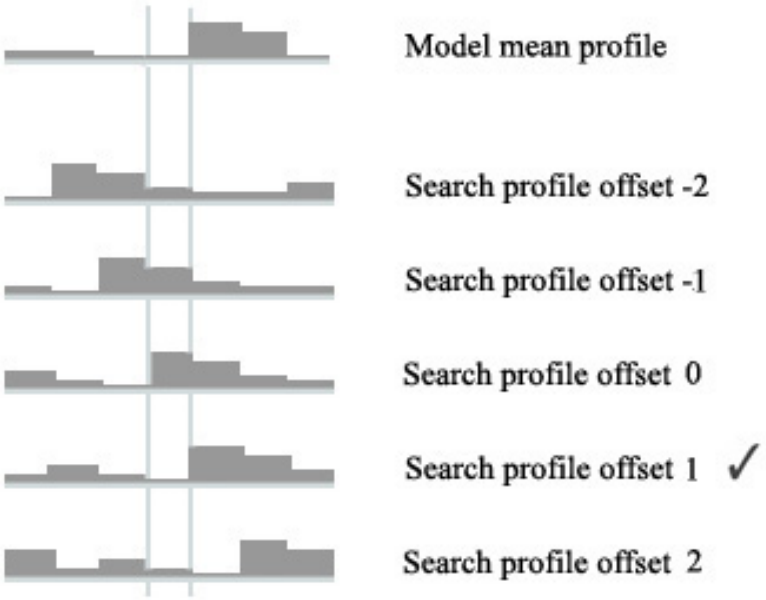


Figure 2.11: *Searching for a profile match.*
The top profile is the model mean profile $\bar{\boldsymbol{g}}$. The remaining profiles are profiles generated during the search by sampling the image around the current position of the landmark. The best match happens to be at offset 1.
In practice, we use the Mahalanobis distance of the normalized intensity gradient rather than the simple skyline match indicated here.

The distance between a search profile $\boldsymbol{g}$ and the model mean profile $\bar{\boldsymbol{g}}$ is calculated using the Mahalanobis distance

$$Distance = (\boldsymbol{g} - \bar{\boldsymbol{g}})^T \boldsymbol{S}_g^{-1} (\boldsymbol{g} - \bar{\boldsymbol{g}}) \ . \tag{2.6}$$

(Some people call this the *squared* Mahalanobis distance, since it simplifies to the square of the euclidean distance when the covariance matrix $\boldsymbol{S_g}$ is an identity matrix.) One of the search profiles will have the lowest distance. It is the position of the center of this profile (which is an offset along the whisker) that is the new suggested position of the landmark.

This process is repeated for each landmark (as shown in step 3 in Figure 2.5) before handing control back to the shape model.

2.5 Multi-resolution search

There is one more wrinkle. Before the search begins, we build an image pyramid, and repeat the ASM search (Figure 2.5) at each level, from coarse to fine resolution. Each image in the image pyramid is a down-scaled version of the image above it (Figure 2.12 overleaf). This

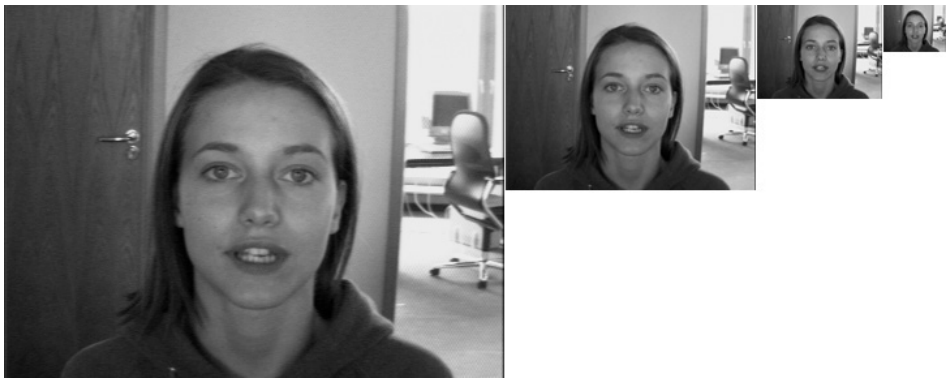


Figure 2.12: *An image pyramid.*
Each image has a quarter of the resolution of the image above it.

is a simple pyramid — more sophisticated pyramids (not used in this project) can be found in [1] and [37].

The start shape for the first search (which is on the coarsest image) is the shape generated from the global face detector. The start shape at subsequent levels is the best face found by the search at the level below.

We need to decide when convergence has been reached so we can move to the next level, or, at the top level, announce the search complete. Techniques for testing convergence will be presented in section 5.4.2.

The shape model is the same across all pyramid levels (apart from scaling), but a separate profile model is needed for each level.

Using this multi-resolution approach is more efficient, more robust, and converges correctly to the correct shape from further away than searching at a single resolution.

Chapter 3

History and Literature Review

This chapter places Active Shape Models (ASMs) in the context of related methods for finding image features, and along the way gives an overview of the literature. This is just a short review — more complete histories can be found in, for example, Hjelmås and Low [42] or Stegmann [65].

In our case, image features are facial features. Finding features in faces has an immediate appeal, but much of the research in this field has been in the context of medical imaging, where features could be vertebrae or the shape of a lung.

The two face detectors used in this thesis are the Rowley [58] and the Viola Jones [67] detectors. Section 4.4 gives details.

3.1 Image segmentation

Basic feature location methods fall under the general category of "image segmentation". A simple approach is to move a canonical or "golden" image over the image looking for matches. Variability of image textures and features make this method inadequate. There are a host of other techniques such as edge and line detection which are described in most image processing text books, for example Gonzalez and Woods [37]. These techniques form the basis of the profile model which is one leg of the ASM.

3.2 Deformable models

Unlike these basic methods, ASMs belong to the class of models which, after a shape is situated near an image feature, interact with the image to warp the shape to the feature. They are *deformable models*.

A well known deformable model is the snake, also called the Active Contour Model [46]. Snakes iteratively minimize the sum of "external energy" and "internal energy". External

energy is minimized as the snake assumes the shape of the desired image feature. Internal energy constrains the snake shape, typically by penalizing curviness.

A problem with snakes is that they can hitch onto the wrong feature. Also, snakes do not like to bend and thus have difficulties locating sharply curved features. As you might expect, people have modified snakes to deal with these issues. The eye locater of Yuille and his colleagues is an an example [68]. They handcrafted a deformable template model which incorporates global information about eyes. The trouble with such an approach is that you have to manually design a new model for each application.

Another approach is to use finite element methods to represent prototype shapes, for example Pentland and Sclaroff [53]. This approach takes a single shape and treats it as if it were made of elastic material. However, the "modes of vibration" used to deform the shape may not represent the actual variations of the feature being modeled.

Staib and Duncan presented a method of modeling based on Fourier descriptors [63]. (Early papers on Fourier descriptors are [54, 69].) Different shapes are generated by varying the parameters of a Fourier series defining the coordinates of the shape perimeter. Unfortunately these models cannot easily describe all shapes, especially sharp curves, and choosing parameters for the Fourier series is tricky.

3.3 Point distribution models

Independently of computerized image analysis, and before ASMs were developed, researchers developed statistical models of shape [30]. The idea is that once you represent shapes as vectors, you can apply standard statistical methods to them just like any other multivariate object. These models learn allowable constellations of shape points from training examples and use principal components to build what is called a *Point Distribution Model*. These have been used in diverse ways, for example for categorizing Iron Age broaches [61].

3.4 Active Shape Models

Ideal Point Distribution Models can *only* deform in ways that are characteristic of the object. Cootes and his colleagues were seeking models which do exactly that — so if a beard, say, covers the chin, the shape model can "override the image" to approximate the position of the chin under the beard. It was therefore natural (but perhaps only in retrospect) to adopt Point Distribution Models. This synthesis of ideas from image processing and statistical shape modeling led to the Active Shape Model.

The first parametric statistical shape model for image analysis using principal components of inter-landmark distances was presented by Cootes and Taylor in [12]. Building on this approach, Cootes, Taylor, and their colleagues, then released a series of papers that cumulated in what we call the classical Active Shape Model [15–17, 21, 22].

Apart from better performance at the time than other facial feature locaters, ASMs had the

advantage that they did not require a handcrafted model, although the training images had to be manually landmarked. ASMs build models by training on a set of reference images, which means that they mechanically extract important characteristics of the training set. In practice, hand tuning is still needed to get best results for a specific application — this thesis is an example.

3.5 Active Appearance Models

Cootes, Edwards, and their colleagues subsequently developed the Active Appearance Model (AAM) [13, 32]. AAMs merge the the shape and texture model into a single model of appearance. Texture is measured across the whole object i.e. the whole face. An optimization technique which pre-calculates residuals during training is used when matching image texture to model texture.

AAMs extend ASMs to a degree that excludes them from this thesis. Nevertheless, we can learn something about ASMs by comparing them to AAMs.

AAMs use the texture across the entire object; ASMs use image texture only in the area of the landmarks — thus ASMs do not use all the available information across the face surface. One can debate whether this is a disadvantage, since landmarks are logically located on areas of interest which carry most important information, and thus ASMs naturally prune irrelevant information.

On the other hand, ASMs search in regions around the landmarks and thus have a larger capture range than AAMs, which only uses the image directly under the current shape.

AAMs need a smaller number of landmarks because of the extra information provided by the texture model.

ASMs for faces can be trained in a few minutes, making it easy to experiment; training AAMs takes much longer.

Cootes et al. [14] report that landmark localization accuracy is better on the whole for ASMs than AAMs, although this may have changed with subsequent developments to the AAM.

3.6 Extensions to ASMs

Many extensions to the classical ASM have been proposed. This section mentions just a few.

Cootes et al. [11, 18] use a shape model which is a mixture of multivariate Gaussians, rather than assuming that the shapes come from a single multivariate Gaussian distribution as in the classical ASM.

Romdhani et al. [57] use Kernel Principal Components Analysis [59] and a Support Vector Machine [8]. This trains on 2D images, but models non-linear changes to face shapes as they are rotated in 3D. The idea is to project non-linearities into a higher-dimensional linear space.

Rogers and Graham [52] robustify ASMs by using robust least-squares techniques to minimize the residuals between the model shape and the suggested shape. A classical ASM, in contrast, uses standard least-squares which is susceptible to outliers.

Van Ginneken et al. [36] take the tack of replacing the 1D normalized first derivative profiles used in the classical ASM with local descriptors calculated from "locally orderless images" [47]. Their method automatically selects the optimum set of descriptors. They also replace the standard ASM profile model search, which is based on a Mahalanobis distance, with a k-nearest-neighbors classifier.

Zhou et al. [70] present the Bayesian Tangent Shape Model. This estimates shape and pose parameters using Bayesian inference after projecting the shapes into a tangent space.

Al-Zubi [2] proposes the Active Shape Structural Model, which combines ASMs with a model of the structural aspects of the object under consideration. He uses them to analyze hand drawn sketches, an important application for user interfaces, in [3].

Li and Ito [48] build texture models using AdaBoosted histogram classifiers.

A fun example of the ASM at work is described in the paper by Chen et al. [9], which presents a scheme to generate computerized sketches of people's faces. It analyzes subjects' faces with an ASM which uses 2D profiles on the eyes and mouth.

The Active Appearance Model mentioned above has many descendants, which could be considered the grand-children of the ASM. A history of these would take us too far afield but an example is the succession of models developed by Cristinacce et al. [23–29].

Chapter 4

Model Building and Evaluation

This chapter looks at general aspects of model building and evaluation before we dive into actual model building in the next chapter.

A "model" is a facial landmark locater embodied in a piece of software and associated parameter files. "Model evaluation" means comparing models by using them to find landmarks in face images. The term "parameter" is used here in a very general sense: it encompasses things such as the values of constants, but also includes different algorithms.

4.1 The face data

In this project we use three manually landmarked datasets. Table A.1 in the appendix lists all landmarks and their names. Table 4.1 on the next page lists some characteristics of the databases.

1. The **AR face database** [50]

   The AR dataset contains about 4000 768x576 color images of 126 faces. The images are frontal views with different expressions and illumination, and with a flat background. The database is free for academic use and cannot in general be reproduced.

   Only a subset of the AR database is used in this project: this is the subset of of 508 images that has been manually landmarked with 22 points [55]. The subset consists of images with a neutral expression, or a smile, or an angry expression (usually very mild), or with the left light on. (These are the sets labeled 1, 2, 3, and 5 in the AR documentation — the faces obscured by sun glasses or scarves are not used.)

   Figure A.2 in the appendix shows a landmarked AR face.

2. The **BioId face database** with FGnet markup [44].

   The BioId dataset consists of 1521 384x286 pixel monochrome images of 23 different subjects. The images are frontal views with a variety of backgrounds and face sizes. The background is an office interior. The faces are manually landmarked with 20 points. The

	AR	BioId	XM2VTS	
Number of landmarks	22	20	68	
Number of images	508	1521	2360	
Faces with eyes found by Rowley	87.2%	75.2%	82.2%	
Faces found by Viola Jones	98.6%	95.7%	99.2%	
Clean faces	155	826	1272	
Clean faces percent of all	30%	54%	54%	
Clean faces found by Rowley	100%	100%	100%	by definition is 100%
Clean faces found by Viola Jones	100%	96.0%	100%	
Mean search time Rowley (ms)	3600	1100	3600	on 1.5 GHz Pentium
Mean search time Viola Jones (ms)	120	70	120	on 1.5 GHz Pentium

Table 4.1: *Database and face detector statistics.*
Note added 29 Oct 2007: table may be slightly inaccurate after recent tweaks to face attributes.

data set is freely down-loadable and it seems that the faces may be reprinted without legal issues. Figure A.3 shows a landmarked BioId face.

All printed faces in this document are from the BioId set except where noted.

3. The **University of Surrey XM2VTS database**, frontal sets I and II [51].

   The XM2VTS frontal image sets contain 2360 720x576 color images of 295 subjects. The pose and lighting is uniform, with a flat background. The faces are manually landmarked with 68 points. The XM2VTS data must be purchased and cannot in general be reproduced. Figure A.1 shows a landmarked XM2VTS face.

4.1.1 Face attributes

Each image is tagged with zero or more of the attributes shown in Table 4.2 on the next page. A few attributes used internally are not shown. See the source file `atface.hpp` for full details. Attribute tags were generated as part of this project and the data will be made publicly available.

Attributes are useful in many ways — for example, we may want to compare the quality of landmark searches on faces with or without eye glasses.

In this document, the term *clean faces* will be used often. *Clean faces* have no attribute bits set.

Filenames in the image databases include a number which specifies the individual in the image. For this project, filenames were extended where necessary to include the individual's number. Thus the BioId file `B0694_08.bmp` is an image of individual number 08, and `B*_08.bmp` specifies all BioId files of that individual.

```
BadImage   0x01  image is bad in some way (blurred, side view of face, etc.)
Glasses    0x02  face is wearing specs
Beard      0x04  beard including possible mustache
Mustache   0x08  mustache
Obscured   0x10  face is obscured (usually by edge of image or subject's hand)
EyesClosed 0x20  eyes closed (partially open is not considered closed)
Expression 0x40  strong expression on face
NnFailed   0x80  Rowley face detector failed (can use as an indication of face "quality")
```

Table 4.2: *Face attributes.*
In the text, the term ***clean faces*** *means faces with no attribute bits set.*

Figure 4.1: *Some BioId faces with their attributes.*

4.1.2 Extending the data by mirroring

We can double the number of landmarked face images by using mirror images. This is not as redundant as it sounds, because lighting effects and small differences in pose and facial symmetry usually make the image texture on one side of the face quite different from the other — in fact, often more so than the texture across two images of the same individual.

Later we will extend the data in another way: by adding extra landmarks (section 5.6).

4.1.3 Issues with the data

The generalization ability of the detector is determined by the range of faces in the training data. From this perspective, the three datasets are not completely satisfactory. The dataset faces are of working adults in the Europe and the USA and exclude children and elderly people. Most of the people in the images are Caucasian. It should also be said that the image sets exclude anyone with unusual looks (very overweight, etc.).

For effective training and testing we need a sufficient number of samples of uncorrelated data. This is not entirely possible with the given datasets because of strong correlations between the images. For example there are only only 23 different individuals in the BioId set, and the

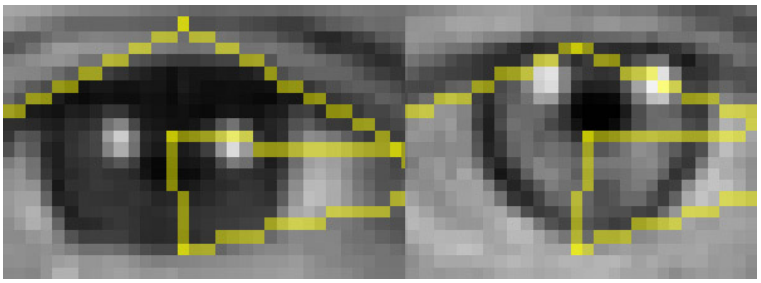


Figure 4.2: *Inconsistent manual landmarks on upper eyelid.*
This is a detail from XM2VTS images 000_3_1 and 001_1_2.

lighting is uniform for most images in each dataset.

Manual landmarking is not a perfect process and a close examination of the data reveals some discrepancies. For example, look at the inconsistency in the position of the landmark on the upper eyelid in the two XM2VTS images in Figure 4.2. (Note also the uniformity of lighting shown by the similarities between the highlights in the two eyes.)

4.2 Partitioning the data

This section discusses how we partition the data in order to estimate the performance of different ASM models. First we review model assessment in more general terms. Hastie et al. [40] chapter 7 gives a more complete description.

The general strategy for selecting parameters is

1. for each model parameter
2. for each parameter value
3. train on a set of faces
4. evaluate the model by using it to locate face landmarks
5. select the value of the parameter that gives the best model
6. test the final model by using it to locate face landmarks.

Two processes are going on here

1. **model selection** which estimates the performance of different models in order to choose one (steps 2-5 above)
2. **model assessment** which estimates the final model's performance on new data (step 6 above).

For both of these processes we need data independent of the training set. We want to measure the *generalization* ability of the model, not its ability on the set it was trained on. Any statistical model captures both systematic structure and random variation in the training

data. To compare two models we want to compare how well they capture systematic structure and ignore random variation. We do this best by testing against independent data. Evaluating against the training data does not separate the random from systematic variation, and will give optimistic results. Furthermore, the test data must be distinct from the validation set so when we adjust model parameters we are not optimizing for the test set.

We thus need three independent datasets:

1. a **training set** for step 3 above
2. a **validation set** for step 4 above
3. a **test set** for step 6 above.

The training set should be the largest, because the training set defines nearly all characteristics of the model. The validation and test sets should be big enough for the variance of the measured results to be acceptable. The unknown ideal sizes for these sets depend on the unknown complexity of the underlying function and amount of data noise. In practice, it would not be unusual to see a size ratio of 8:1:1 for the training:validation:test sets.

There are analytic methods (Akaike Information Criteria and the like) that approximate the validation step using just the training data ([40] section 7.5). These methods require likelihood estimation and are not used in this project.

We also do not use N-fold cross validation, because there is enough raw data, and cross validation makes model building very time consuming ([31] section 9.6).

We now look at two (of the many) possible ways of generating these three sets from the available face data.

4.2.1 Partitioning the data: a not-so-good way

This project started with just the XM2VTS set; the other two databases were discovered later by web searches. The initial strategy used just the clean faces in the XM2VTS set and divided them into into three fixed subsets: a training set (1172 faces), a validation set (50 faces), and a test set (50 faces). *Clean* faces have no face attribute bits set (Table 4.2). The idea at the time was to limit this project to clean faces; later it was realized that this was needlessly restrictive.

The problem with this approach is there is duplication between the training and test sets (and the same is true for the validation set). Although the same images do not appear in both sets, there are some images in both sets of the same face, photographed under identical conditions. This amounts to replication of data. Testing against the test set therefore does not actually test the generalization capability of the classifier.

Another problem is that because of the limited amount of data, the validation and test sets are quite small. Measurements on a small dataset exhibit high variance, and say more about the random nature of the set than the differences between one model and another.

4.2.2 Partitioning the data: a better way

We work around the problems mentioned above by training on the XM2VTS set, validating on the AR set, and doing final testing on the BioId set. The XM2VTS set is used for training because it has the largest number of images and the largest number of landmarks (section 5.1). Using the BioId set for testing allows us to compare results against other published results in Chapter 6.

This strategy is better but not perfect. When a model goes through a long series of refinements based on measurements against a fixed validation set, it is in effect training on the validation data. In this case, we are optimizing the model for the AR data. We minimize this overfitting by using different subsets of the AR data at different times. Most of the tests in this document are done against a set of 200 images randomly sampled without replacement from the AR dataset.

4.3 Software

Software was built to evaluate the techniques in this dissertation. The software itself is not part of the thesis *per se*, although this document occasionally refers to source files. The software and associated data are available at `www.milbo.users.sonic.net/stasm`.

For the record, the code is written in C++ for Microsoft Visual C 6.0. A GNU gcc port is underway. The code uses the GNU Scientific Library [34], with an added class to allow easy manipulation of matrices. The R language [66] was used for analysis of results.

4.4 The face detectors

The ASM search starts with a start shape: the start shape is the model mean shape aligned to the overall position and size of the face found by a global face detector. This project uses two global face detectors:

1. The **Rowley detector** [58] uses a committee of neural networks. The Rowley detector provide eye positions in addition to the face position. We will see that it leads to better landmark searches (section 6.4) although it is has a worse detection rate and is slower than the Viola Jones detector. This thesis uses code from Henry Rowley's web site [58].

2. The **Viola Jones detector** [67] uses a method which accumulates the results of many weak classifiers, each of which looks for a very simple image feature. This thesis uses the OpenCV implementation [56] which is based on a Viola Jones detector extended by Lienhart and Maydt [49].

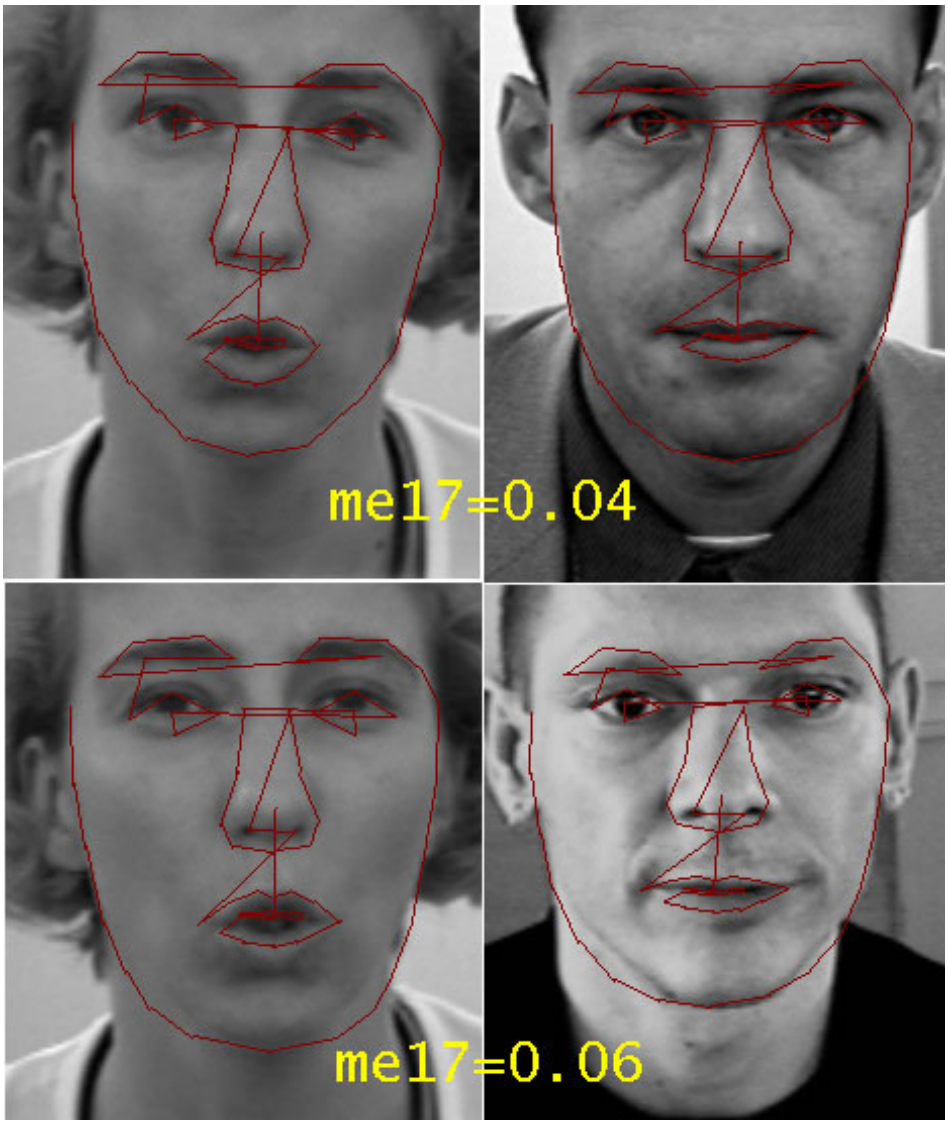


Figure 4.3: *Examples of the me17 measure.*

The OpenCV Viola Jones settings used are the "slow but accurate" settings:

```
SCALE_FACTOR             = 1.1
MIN_NEIGHBORS            = 3
DETECTOR_FLAGS           = 0
MIN_FACE_WIDTH           = ImageWidth/4  (reduce number of false positives)
RETURN_MOST_CENTRAL_FACE = 1
```

```
me17   15,18,21,24,27,29,31,32,34,36,46,47,48,51,54,57,67
Face   0...67
Jaw    0...14
Nose   37...47, 67
Mouth  48...66
Eyes   27...36
LEye   27...31
REye   32...36
```

Table 4.3: *Landmarks used in me17 and other fit summaries.*
The landmarks are numbered using the XM2VTS scheme in Table A.1

4.5 The me17 measure

Results in this document are presented in terms of the *me17* measure. Following Cristinacce [23] section 6.1.4, the me17 is calculated as follows:

1. Average the euclidean distances between each of 17 points located by the search and the corresponding manually landmarked point. The 17 points are the internal BioId landmarks (see Table 4.3).
2. Divide the result by the distance between the eye pupils. In this document, the inter-eye distance is calculated from the reference image (not from landmarks located by the search).
3. If there is more than one image, take the mean me17 over all images.

Figure 4.3 on the previous page shows examples of specific me17s.

By using the me17 common subset of points, measurements can be made on all three databases. We rely on the fact that the images are landmarked in the same way across the three databases. The measure is to some extent arbitrary (it ignores points on the face perimeter and gives more weight to faces with close set eyes, for example), but is about as good as any other for measuring model quality. The me17 is a sum-of-squares criterion, thus minimizing it chooses the maximum likelihood model if the errors are independent and identically normally distributed.

Another approach is to take the median instead of the mean in step 3 above. We use means in this document because we want to see the degradation in total fit caused by outliers (medians are less affected by outliers). The me17 distributions tend to have long right tails, consequently the median is usually less than the mean.

All results in this document are presented in terms of the me17 measure, unless otherwise stated. On occasion, more detailed measurements are needed. An example would be `Eyes` which measures the fit on the eyes only (Table 4.3).

```
Parameters are param1, param2

Model   AR   BioId  XM2VTS  Time
Ref    0.0     0.0     0.0   0
17,1   4.1    -2.0    -1.7   1
17,2  -2.5    -5.5    -5.2   2
17,3  -4.0    -7.0    -5.3   1   <chosen, lowest AR
17,4  -2.4    -8.6    -2.0  -3
...
```

Table 4.4: *Example evaluation table.*
The table shows performance as a percentage relative to a reference model.

4.6 Tables for comparing models

Evaluation results in the next chapter are presented in the format shown in Table 4.4.

The very top line specifies which parameters are being adjusted — in this case `param1,param2`. The parameters will be discussed in the text accompanying each table, and in general are defined in the source files `masmconf.hpp` and `mascmconf.cpp`.

The `Model` column specifies the values of the parameters. Thus the entry `17,1` means that `param1=17` and `param2=1`.

The filetype headings refer to the three datasets (AR, BioId, XM2VTS). In practice, we will display only one or two filetypes, not all three as in the example.

The reference model is usually the first row of the table. All its entries are zero.

The cell entries are percentages and give me17 performance relative to the reference model, averaged across all images. More precisely each cell is
`100% * (mean_me17 - mean_reference_me17) / mean_reference_me17`.
Since this is an error measurement, good models have low values. Since it is a relative measure, values that are better than the reference are negative.

An optional `Time` column measures search time relative to the time taken by the reference model, in percent. This measures search time once the global face detector has found the face i.e. the time does not include the time taken to find the face. Once again, this is a relative measure, so times faster than the reference time are negative.

The `<chosen` note, often with a brief comment, specifies which model was chosen.

4.7 Graphs for comparing models

The table format described in the previous section is good for concise comparison of many models. Often we want more details though, and use graphs. A standard way to do this is to plot an Empirical Cumulative Distribution Function (ECDF, conveniently pronounced "eck diff"). Given a vector of me17 error measurements `errs`, the pseudo code (in actual fact, R code [66]) to plot an ECDF is

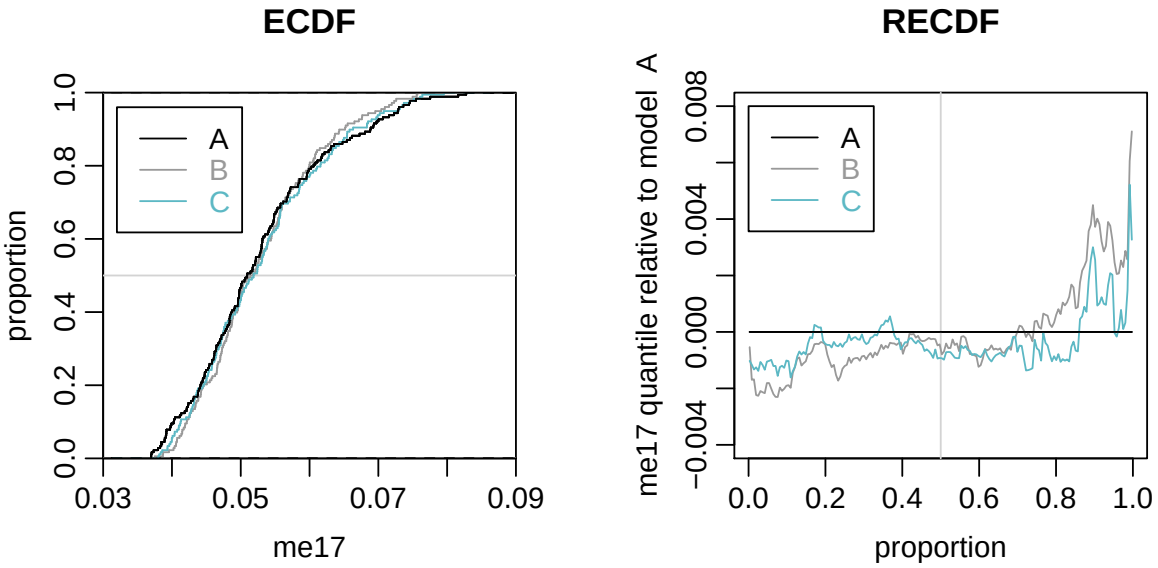


Figure 4.4: *Graphs for comparing models.*
Left *cumulative distributions of the fit for hypothetical models A, B, and C.*
Right *the relative cumulative distributions for the same three models.*

```
x <- sort(errs)
y <- (1:length(errs)) / length(errs)   # length returns the number of of elements
plot(x, y).
```

The code is repeated for each model to give a graph like that on the left side of Figure 4.4. The quicker the S curve starts and the faster it reaches the top, the better the model. The horizontal gray line shows the median values. In the example graph, the probability of getting an me17 of say 0.05 or less is about 0.42. Overplotting may make it hard to compare similar models in ECDF plots.

Relative Empirical Cumulative Distribution Function (RECDF, "reck diff") plots focus on the relative performance of the models and reduce overplotting. See the right side of Figure 4.4. Here we plot quantiles *relative* to a reference model, in this case model A. Given vectors of measurements `ref_errs` and `errs`, the code to plot a RECDF is

```
assert(length(ref_errs) == length(errs))
x <- (1:length(errs)) / length(errs)
y <- sort(ref_errs) - sort(errs)
plot(x, y).
```

The code is repeated for each model except the reference model. The graph for the reference model is merely a horizontal line (black in the figure). The vertical gray line shows the median values. The higher a model's curve, the better the model.

In the example, models B and C start off badly but end up being better than A. In other

words, B and C's best case performance is worse on the whole than A; B and C's worst case performance is better than A. Given this, and the fact that C has the lowest median value, C would be considered the best model in this example. But the models are very similar and, taking into account measurement variance, there is not much to choose between them.

I have not seen RECDFs described elsewhere, but they are surely a reinvention. They are similar in spirit to QQ plots, except that the reference distribution is drawn as a horizontal line (for QQ plots, see for example [45] section 4.6). They are also related to Tukey mean-difference plots but RECDFs allow more than two models to be compared ([10] section 2.2 or [6]).

4.8 Hazards of parameter selection

Generally, for each parameter we select the parameter value that yields the lowest mean me17 on the AR validation set. This holds especially if the lowest me17 is at the bottom of a basin of low values. If models are similar then other factors come into play, like complexity and execution speed.

Statistical fluctuation may cause misleading results. A model may appear to be better than another model just because it happens by chance to have a lower error on that particular validation set. Thus when we say that we choose such and such a parameter value because it results in the lowest error, what we are really saying is that the odds are that the parameter value results in a better model. A formal statistical approach (not used in this project) would take into account the variance of the measurements, typically with a paired t-test. The standard formal approach is "don't accept a value unless it is statistically significant", which may throw away potentially good results. On the other hand, not using the standard approach increases the chance that we accept a parameter value or procedure that in fact does not improve the model.

The best setting for a parameter depends on the context. If other parameters are changed then the best setting for the parameter could change. We are optimizing a model with many parameters, with an error hyper-surface in many dimensions. It is not feasible to find the global error minimum. The strategy in this project is to move forward in a sequence of build-evaluate steps, adjusting a new parameter at each step (section 4.2). This is essentially greedy descent, with descent directions constrained to the axes of the parameter space. The best we can hope for is that our model building exercise finds a good minimum. It is unlikely to be the best. This is not to argue that this is a bad strategy, but that we should be aware of its limitations.

In practice, to sort the wheat from the chaff, lots of experiments were tried in "random" order to build a good initial model. In the next chapter we then start again. Using the initial model as a base, we will build and document a model using the parameter evaluation sequence laid out in the chapter. A different evaluation sequence would result in somewhat different parameter values.

4.9 Summary

The points to remember are:

1. We will train on the XM2VTS set, evaluate on the AR set, and test on the BioId set (section 4.2.2).
2. Results will be presented as me17s and times relative to a reference model, expressed as a percentage (section 4.6).

Chapter 5

Model Parameters and Variations

A thousand things advance;
nine hundred and ninety-nine retreat:
that is progress.
Henri Frederic Amiel [4]

The previous chapter described the general approach to model building used in this project. Now we systematically work through some variations and parameters of the Active Shape Model. Readers who are not interested in the details of parameter selection may go directly to Chapter 6, where all the important results are summarized.

We will train on the XM2VTS set and evaluate against the AR set, with a few exceptions, as motivated in section 4.2.2. The format of the graphs and tables in this chapter was described in sections 4.6 and 4.7. It is important to remember that the figures in the tables are me17s and times relative to a reference model, expressed as a percentage (section 4.6).

5.1 Number of landmarks

A straightforward way to improve the fit is to increase the number of landmarks in the model. The error decreases approximately quadratically as the number of landmarks increases.

Figure 5.1 on the next page shows fit versus number of landmarks. The vertical axis is the me17, normalized so the error is 1 for 68 points. The black line is a loess fit with `degree=2` and `span=0.75` [10]. The blue line is a quadratic fit.

We pay a price in search time because search time is proportional to the number of landmarks. But perhaps we should be grateful that time increases only linearly with the number of landmarks — for most extensions to the ASM, search time is worse than linear in the number of landmarks (e.g. [32, 36]).

All models up to section 5.6 in this chapter are built using all 68 XM2VTS points. So we

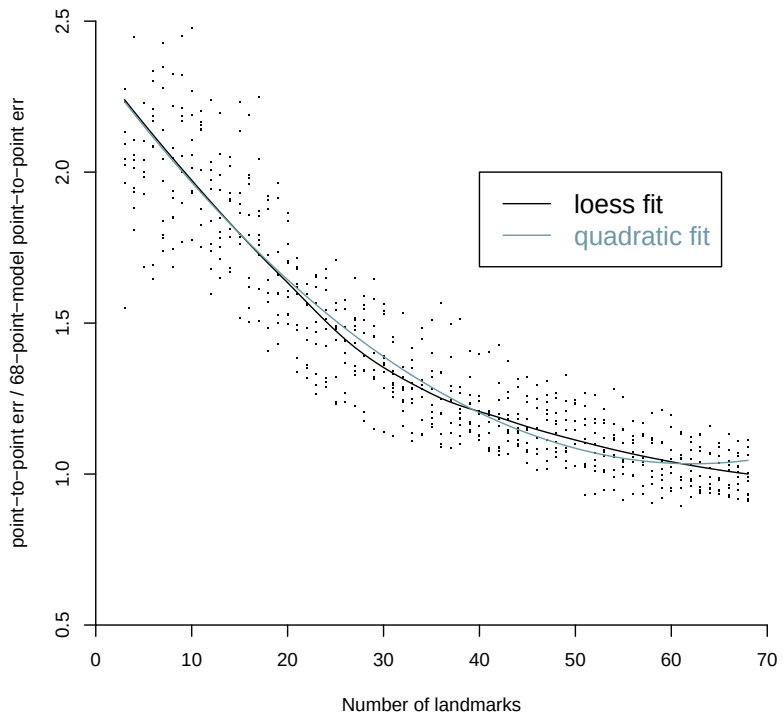


Figure 5.1: *Number of landmarks.*

will train and fit models with 68 points but use just the subset of seventeen me17 points for evaluation. In section 5.6 we add more landmarks, and thereafter use an 84 point model.

5.1.1 Details

Each point on the graph was generated from a classical 1D ASM, built by randomly choosing the given number of landmarks from the 68 available in the XM2VTS set. This was repeated 10 times for each value of `nLandmarks`, to give 10 points for each value of `nLandmarks`. For a given number of landmarks, certain landmark sets are better than others, causing scatter among the 10 points. The decrease in scatter as the number of landmarks increases is expected because the total number of observations per graph point increases as the number of landmarks increases.

In detail, the graph was created as follows:

```
TrainSet = a random selection of 1200 clean faces in the XM2VTS set
TestSet = remaining clean faces in the XM2VTS set (72 faces)
for nLandmarks = 3 to 68
    repeat 10 times
       LandmarkSet = randomly select nLandmarks landmarks in TrainSet
       train on LandmarkSet
       measure mean fit on TestSet --- result is one point on the graph
```

Model	AR	XM2VTS	
AlignToRefShape	0.0	0.0	
Face	4.3	7.0	
FaceAndEyes	0.7	7.4	
FaceAndEyes_RealignEyes	-0.7	5.8	<chosen

Table 5.1: *Method used to generate the start shape.*
The figures are relative me17s expressed as percentages (section 4.6).

5.2 Generating the start shape

Before the ASM search begins, we use a global face detector to locate the approximate position and size of the face. We then generate a start shape by using this position and size to position and scale the model mean face. For now we confine ourselves to the Rowley face detector (section 4.4).

The quality of the start shape is important. It is is unlikely that an ASM search will recover completely from a bad start shape.

Four methods of generating the start shape are considered:

1. **AlignToRefShape** Align the mean shape to the reference (i.e. the manually landmarked) shape.
   For real searches the reference shape is not available, but during testing this method provides a reference to compare the methods below.

2. **Face** Scale and align the mean shape to the face position determined by the face detector.

3. **FaceAndEyes** Scale and align the mean shape to the face and eye positions determined by the face detector.

4. **FaceAndEyes_RealignEyes** Start off as above (FaceAndEyes) to generate an initial StartShape. Then generate another start shape, StartShapeEyes, by aligning the mean shape to the eye position found by the face detector (ignoring the overall face position found by the face detector). If the inter-eye distance in StartShapeEyes is very different from StartShape, then blend the eyes in StartShapeEyes into StartShape. Use the resulting StartShape as the start shape. See `startshape.cpp` for details.
   The net effect is to use eye positions to correct start shapes that are too big.

Table 5.1 gives measurements and shows that the best results are obtained from FaceAndEyes_RealignEyes. The AR figure for the chosen model is an anomaly because it is less than the figure for AlignToRefShape, a supposedly impossible-to-beat number. This is presumably due to statistical variation.

Model	AR	XM2VTS	
AdHocScaling	0.0	0.0	
width 140, nearest-pixel	16.4	1.8	
width 140, bilinear	11.5	2.9	
width 160, nearest-pixel	13.5	1.0	
width 160, bilinear	11.3	0.1	
width 180, nearest-pixel	1.9	1.2	
width 180, bilinear	0.0	0.0	<chosen, lowest AR
width 200, nearest-pixel	2.5	0.4	
width 200, bilinear	1.3	0.2	
width 220, nearest-pixel	1.1	1.1	
width 220, bilinear	1.2	1.7	
width 240, nearest-pixel	1.3	0.8	
width 240, bilinear	2.2	-0.2	

Table 5.2: *Scaled face width, 1D profiles.*
Here we prescale all faces to the given width before training and searching (except for AdHocScaling)

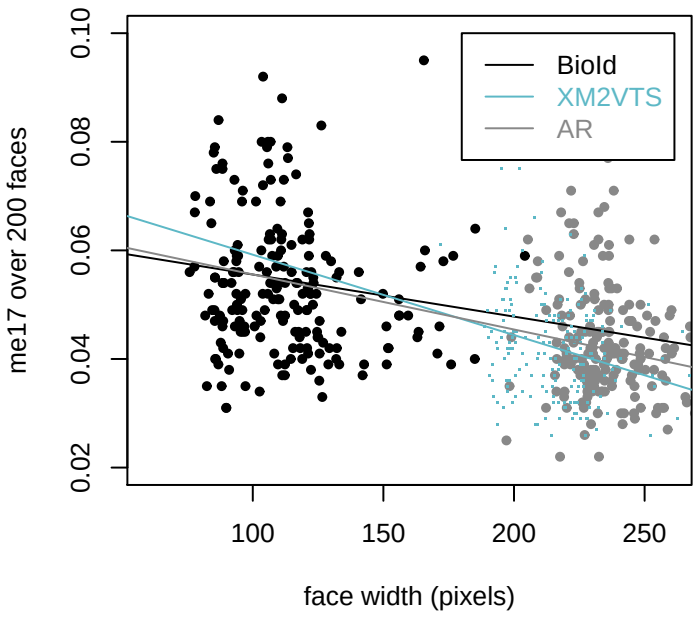


Figure 5.2: *Face width, with regression lines. 1D profiles.*

5.3 Face size scaling

Profile models would work better if all faces were the same size. Reality is not so kind, but we can instead prescale the faces to a common pixel width before using them. Table 5.2 on the previous page shows the results.

The reference model **AdHocScaling** was the first implementation for this project. This does no rescaling during model building. During search, it doubles the size of the face if the width is less than 120 pixels. If the width of the face is greater then 350 it resizes the face to 280 pixels. The idea is to minimize the amount of non-integral scaling of the image. This method appears effective on the available data but was rejected because it is too messy.

The subsequent models simply prescale the faces to a fixed width before both model building and searching. We scale using either nearest-pixel or bilinear interpolation. The best results are with bilinear scaling to a width of 180 pixels.

Figure 5.2 on the previous page shows me17 plotted against face width. The linear regression lines show that the fit improves as original image face size increases, within the range of widths in the databases.

5.4 Models with 1D profiles

This section looks at the several basic parameters of the classical ASM using one-dimensional (1D) profiles. 1D profiles were described in section 2.4.1.

5.4.1 Profile model search parameters: nProfWidth and nPixSearch

During profile matching, we search in the vicinity of each landmark for the best profile match. In this section we look at two parameters:

1. **nProfWidth** is the number of pixels in the model profile for each 1D landmark. This width is always an odd number: the middle pixel plus an equal number of pixels on each side of the center pixel. For example, `nProfWidth=11` means a middle pixel and 5 pixels on each side.
2. **nPixSearch:** During search, we sample the image at displacements of `-nPixSearch` to `+nPixSearch` pixels along the whisker

Table 5.3 on the next page gives measurements.

Parameters are nProfWidth, nPixSearch

Model	AR	XM2VTS	
Ref	0.0	0.0	
9,1	8.6	3.2	
9,2	4.9	1.8	
9,3	4.2	2.5	
9,4	8.7	2.9	
9,5	16.2	9.5	
9,6	16.2	11.1	
9,7	27.1	17.5	
11,1	4.9	1.2	
11,2	0.0	-0.2	
11,3	1.4	-0.8	
11,4	1.2	3.1	
11,5	5.2	5.0	
11,6	14.5	7.7	
11,7	15.1	13.9	
13,1	4.3	-0.5	
13,2	-1.3	-3.3	
13,3	-0.2	-2.3	
13,4	0.1	-0.1	
13,5	4.9	0.9	
13,6	14.5	4.2	
13,7	18.5	6.6	
15,1	4.2	-1.7	
15,2	-2.0	-4.6	
15,3	-0.8	-4.2	
15,4	0.1	-2.2	
15,5	4.7	0.4	
15,6	10.7	2.5	
15,7	32.9	3.6	
17,1	4.1	-1.7	
17,2	-2.5	-5.2	
17,3	-4.0	-5.3	<chosen, lowest AR
17,4	-2.4	-2.0	
17,5	3.3	0.5	
17,6	11.7	0.5	
17,7	52.7	1.5	
19,1	1.2	-3.3	
19,2	-3.0	-5.5	
19,3	-2.4	-5.4	
19,4	-1.6	-3.1	
19,5	3.4	-2.1	
19,6	8.9	-0.4	
19,7	45.8	0.0	

Table 5.3: *nProfWidth and nPixSearch, 1D profiles.*

5.4.2 Search parameters: nMaxSearchIters and nQualifyingDisplacements

During profile matching, we iterate for a solution (Figure 2.5). The profile match at each landmark suggests a displacement from the current position. We declare convergence and stop iterating when either of the following conditions is met:

1. **nMaxSearchIters** is reached.
2. **nQualifyingDisplacements** of the landmark displacements are less than or equal to the maximum displacement nPixSearch. Cootes and Taylor [20] section 7.3 proposes a slightly different criterion: the number of landmark displacements that are within 50% of nPixSearch. Using a different criterion here allows the same code to be used for both 1D and 2D profiles:
   `ABS(xDisplacement) + ABS(yDisplacement) <= nPixSearch`.
   This is city block distance. For 1D profiles, yDisplacement is always 0. Note added 29 Oct 2007: the above description is slightly different to the actual code, see nGoodLandmarks in stasm/asm.cpp.

Table 5.4 on the next page gives measurements. We see that good results can be obtained with `nMaxSearchIters=4` and `nQualifyingDisplacements=80`.

Parameters are nMaxSearchIters, nQualifyingDisplacements

Model	AR	XM2VTS	
Ref	0.0	0.0	
2,75	2.3	3.5	
2,80	2.3	2.6	
2,85	2.4	1.9	
2,90	2.2	1.8	
2,95	2.3	1.5	
3,75	-0.9	0.5	
3,80	-1.6	0.9	
3,85	-1.0	0.8	
3,90	-1.3	0.5	
3,95	-1.5	-0.1	
4,75	-2.3	0.2	
4,80	-2.4	0.5	<chosen, lowest AR
4,85	-1.7	0.0	
4,90	-1.7	-1.0	
4,95	-1.7	-0.8	
5,75	0.0	-1.1	
5,80	-0.4	-0.7	
5,85	-1.1	-1.0	
5,90	-0.3	-1.5	
5,95	-0.6	-1.4	
6,75	-0.5	0.1	
6,80	-0.6	-0.1	
6,85	-0.3	-0.2	
6,90	-0.8	-1.0	
6,95	-0.4	-0.6	
7,75	-0.5	-0.3	
7,80	0.3	-0.9	
7,85	0.8	0.0	
7,90	0.3	0.2	
7,95	0.1	-0.2	

Table 5.4: *nMaxSearchIters and nQualifyingDisplacements, 1D profiles.*

5.4.3 Shape model parameters: nEigs and bMax

During the landmark search, we generate a suggested shape by profile matching and then conform the suggested shape to the shape model.

For best results we need to choose the appropriate `nEigs` and `bMax` parameters for the shape model. `nEigs` is the number of eigenvectors used by the shape model. `bMax` is a parameter controlling the maximum allowable value of elements of $\boldsymbol{b}$ in equation 2.2.

In the rectangle example in section 2.3.2 there were eight eigenvectors. But we needed only two eigenvectors to describe any of the shapes i.e. `nEigs` was two in that example. It was clear that two eigenvectors were sufficient because there were only two large eigenvalues. For face shapes there are 136 eigenvectors (136 = number of x- and y-coordinates for 68 landmarks). The magnitude of the eigenvalues decreases relatively smoothly and there is no obvious cut-off point. Thus we choose `nEigs` by testing.

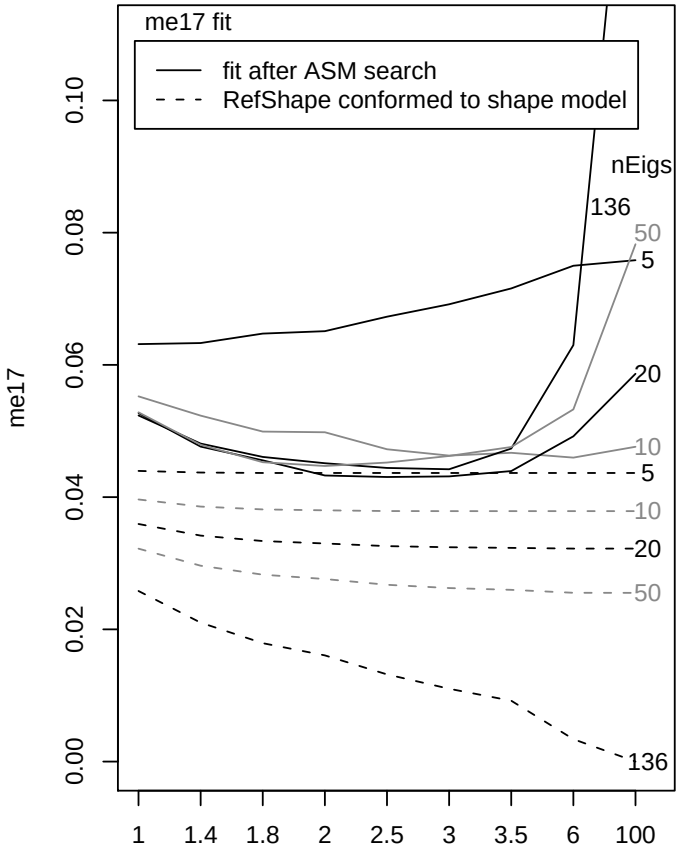
Figure 5.3(a) on the next page shows the results. This graph was generated by applying hard limits to each element b_i of $\boldsymbol{b}$ i.e. clipping them to $\pm bMax * sqrt(\lambda_i)$. Note that the increments along the x-axis are not uniform.

We can get an idea of the lower-bound possible for any search by taking a manually landmarked face shape and conforming it to the shape model. Ideally there should be no change because we should not change a face shape that is already valid. We can get close to this ideal if we use a large `bMax` and a full set of eigenvalues. But using such a flexible shape model allows bad profile matches to pass unchecked and so does not give the best results when searching. This can be seen by the increase on the right of the solid curves.

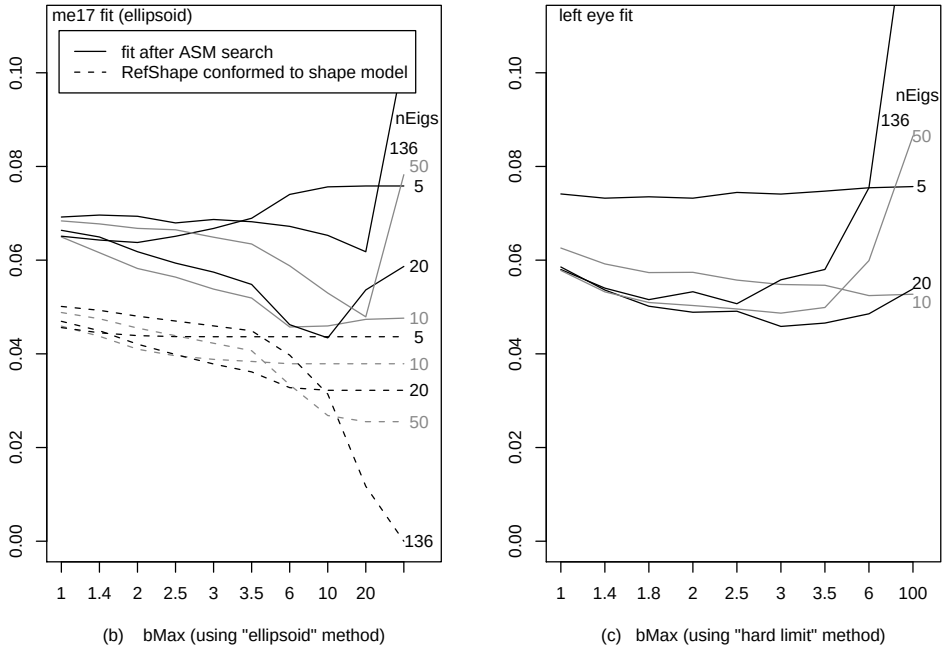
A variation [20] on the method described above is to constrain $\boldsymbol{b}$ to be in a hyper-ellipsoid i.e. scale the b_is so that $sqrt(\sum b_i^2/\lambda_i) < bMax$. We force the inequality when necessary by multiplying each b_i by $bMax/sqrt(\sum b_i^2/\lambda_i)$. A comparison of subfigure (b) to subfigure (a) shows that this slightly more complicated variation does not give better results (for the given code and databases) even though it has stronger theoretical justification, and so is not used in this project.

Subfigure (c) shows results for the same images as subfigure (a), but displays the results for the left eye instead of me17. A comparison of the graphs shows that the best parameter values for the eye are different from the best values for me17: the eye prefers a more flexible model (higher `nEigs` and `bMax`). In spite of this, this project does not do any special shape model processing for eyes or other features.

Table 5.5 on page 47 shows results for the AR dataset at a finer resolution. The chosen values of `nEigs=20` and `bMax=1.8` give the best results for the AR dataset.



(a) bMax (using "hard limit" method)



(b) bMax (using "ellipsoid" method)

(c) bMax (using "hard limit" method)

Figure 5.3: nEigs and bMax, 1D profiles.
Gray and black are used to help distinguish curves and have no special meaning.

Parameters are nEigs, bMax

Model	AR	XM2VTS	
Ref	0.0	0.0	
12,1.0	0.0	0.0	
12,1.2	-3.5	-3.5	
12,1.4	-6.9	-1.6	
12,1.6	-6.5	-4.9	
12,1.8	-6.8	-2.2	
12,2.0	-6.1	-3.0	
12,2.2	-7.8	-2.5	
12,2.6	-3.9	-1.5	
12,3.0	-5.9	-0.9	
12,5.0	0.3	0.9	
14,1.0	0.5	-1.2	
14,1.2	-4.2	-2.9	
14,1.4	-4.3	-3.4	
14,1.6	-5.2	-6.4	
14,1.8	-7.0	-4.9	
14,2.0	-7.7	-2.9	
14,2.2	-7.0	-3.1	
14,2.6	-6.2	-3.6	
14,3.0	-3.9	-1.4	
14,5.0	2.4	-0.6	
18,1.0	-1.6	-2.1	
18,1.2	-4.2	-4.4	
18,1.4	-5.5	-4.9	
18,1.6	-7.9	-4.4	
18,1.8	-6.8	-4.5	
18,2.0	-6.4	-4.0	
18,2.2	-6.0	-3.4	
18,2.6	-7.1	-1.7	
18,3.0	-3.6	1.1	
18,5.0	5.4	2.5	
20,1.0	-1.6	-2.4	
20,1.2	-3.9	-4.7	
20,1.4	-6.4	-6.9	
20,1.6	-6.5	-4.7	
20,1.8	-9.0	-5.1	<chosen, lowest AR, good XM2VTS
20,2.0	-8.0	-4.8	
20,2.2	-6.8	-3.3	
20,2.6	-6.6	-2.2	
20,3.0	-3.0	0.0	
22,1.0	-3.1	-2.8	
22,1.2	-6.1	-5.5	
22,1.4	-6.9	-6.9	
22,1.6	-8.9	-4.8	
22,1.8	-8.6	-3.8	
22,2.0	-9.0	-3.1	
22,2.2	-7.5	-3.7	
22,2.6	-5.1	-1.5	
22,3.0	-3.1	0.4	

Table 5.5: *nEigs and bMax ("hard limit" method), 1D profiles.*

```
Parameters are ShapeNoise (pixels), xRhsStretchShape (rhs face stretch)

File   Model      me17
Ref    0.75,1.08  0.0
AR     0,0        2.1
XM2VTS 0,0        1.6
```

Table 5.6: *Fit improved by adding noise to the training shapes, 1D profiles.*

5.4.4 Adding noise during training

A shape model built by adding noise to the input shapes during training gives slightly improved fits. Noise is added as follows:

1. Displace each landmark by a random amount. In effect, this increases variability in the training set face shapes.
2. Stretch or contract just the right hand side of the face by a random amount. This is done by multiplying the x position (relative to the face center) of each right-side landmark by a random number near 1. Stretching possibly helps because the faces in the XM2VTS training set are mostly facing the camera directly. In the AR set, the faces, although frontal, are sometimes facing slightly away from the camera, which makes the face less symmetrical.

Table 5.6 above shows the fit on model built without noise, versus a model built by adding gaussian noise. The AR me17 improves by about 2%.

5.4.5 Number of pyramid levels

The search proceeds through the image pyramid, from a low resolution image to the original image resolution. How many pyramid levels are best? Table 5.7 below shows some results. Three pyramid levels produces a slightly better me17 for the AR set, but the jaw fit is far worse, so it seems best to stay with the 4 pyramid levels of the classical ASM.

At the other end of the pyramid, terminating a search at a resolution less than the original gives worse results, perhaps obviously.

```
File   Model              Face  me17   Jaw  Nose Mouth  Eyes

AR     nPyramidLevels=4    0.0   0.0   0.0   0.0   0.0   0.0
AR     nPyramidLevels=3    1.3  -0.3  14.5  -0.6   8.4   1.5

XM2VTS nPyramidLevels=4    0.0   0.0   0.0   0.0   0.0   0.0
XM2VTS nPyramidLevels=3    2.2   2.1   2.6   1.2   2.1  -1.1
```

Table 5.7: *Number of pyramid levels, 1D profiles.*

Model	AR	XM2VTS	Time
Reference	0.0	0.0	0.0
Bilinear	-2.4	-0.7	17.7
NearestPix	-2.6	-0.8	1.4

Table 5.8: *Method used to scale the image, 1D profiles.*
This is the first table in this chapter with a Time column. Times are relative to the reference model (see section 4.6).

5.4.6 Method of scaling images

We form the image pyramid by scaling the image. Table 5.8 above shows results from scaling using bilinear versus nearest-pixel resampling. Nearest-pixel is faster and gives results that are almost the same as those for bilinear.

5.4.7 Pyramid ratio

Each image in the image pyramid is a scaled version of the image above it. Table 5.9 below shows results using different scale factors. A scale factor of 2 is best, probably because the pixels are scaled by an integer, which tends to preserve image texture. (Scaling by 3 also scales by an integer, but scales by too much.)

Model	AR	XM2VTS	
sqrt2	7.1	-3.6	
sqrt3	12.8	-1.8	
2	0.0	0.0	<chosen, lowest AR
3	12.1	1.1	

Table 5.9: *Pyramid ratio, 1D profiles.*

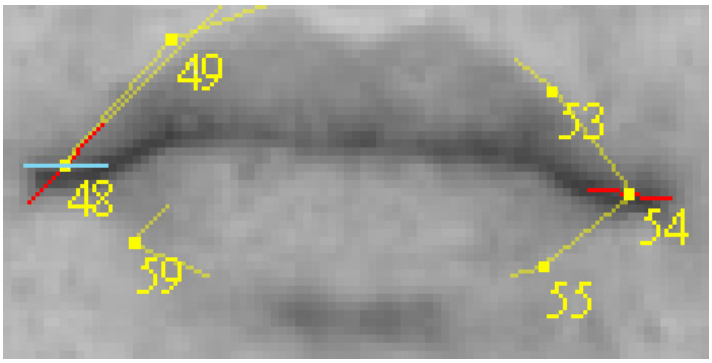


Figure 5.4: *Whisker directions for some XM2VTS landmarks.*
The red whiskers are from the classical ASM. The pale blue whisker uses explicitly specified "previous" and "next" landmarks.

File	Model	Face	me17	Jaw	Nose	Mouth	Eyes
AR	UseLandmarkOrdering	0.0	0.0	0.0	0.0	0.0	0.0
AR	UseExplicitSpec	-2.5	-2.2	-4.3	-9.0	-2.9	0.3
XM2VTS	UseLandmarkOrdering	0.0	0.0	0.0	0.0	0.0	0.0
XM2VTS	UseExplicitSpec	-0.4	0.6	0.1	-2.6	-1.6	3.0

Table 5.10: *Whisker directions, 1D profiles.*
This compares results using whisker directions determined by ordering of landmarks (i.e. classical ASM), versus results using explicitly specified whisker directions.

5.4.8 Whisker directions

In the classical ASM, the direction of a whisker is determined by the order in which the landmarks are numbered. This is because the whisker direction is determined by the position of previous and next landmarks relative to the current landmark. This is rather arbitrary. Consider landmark 48 in Figure 5.4, which shows selected XM2VTS landmarks. The previous landmark is 47, which is the right nostril. The next landmark is 49 which is on the upper lip. Compare the odd direction of the resulting whisker (shown in red) to that of the whisker on the symmetrical landmark 54.

Instead of using the ordering of previous and next landmarks to determine whisker direction, we can explicitly specify the landmarks to be used. In Figure 5.4, we use landmarks 59 and 49 for this purpose, and the resulting whisker is shown in pale blue.

Table 5.10 shows some results. The overall results for the AR set are slightly better when we explicitly the "previous" and "next" landmarks.

Note that none of this applies to 2D profiles which are not rotated for alignment to the shape edges.

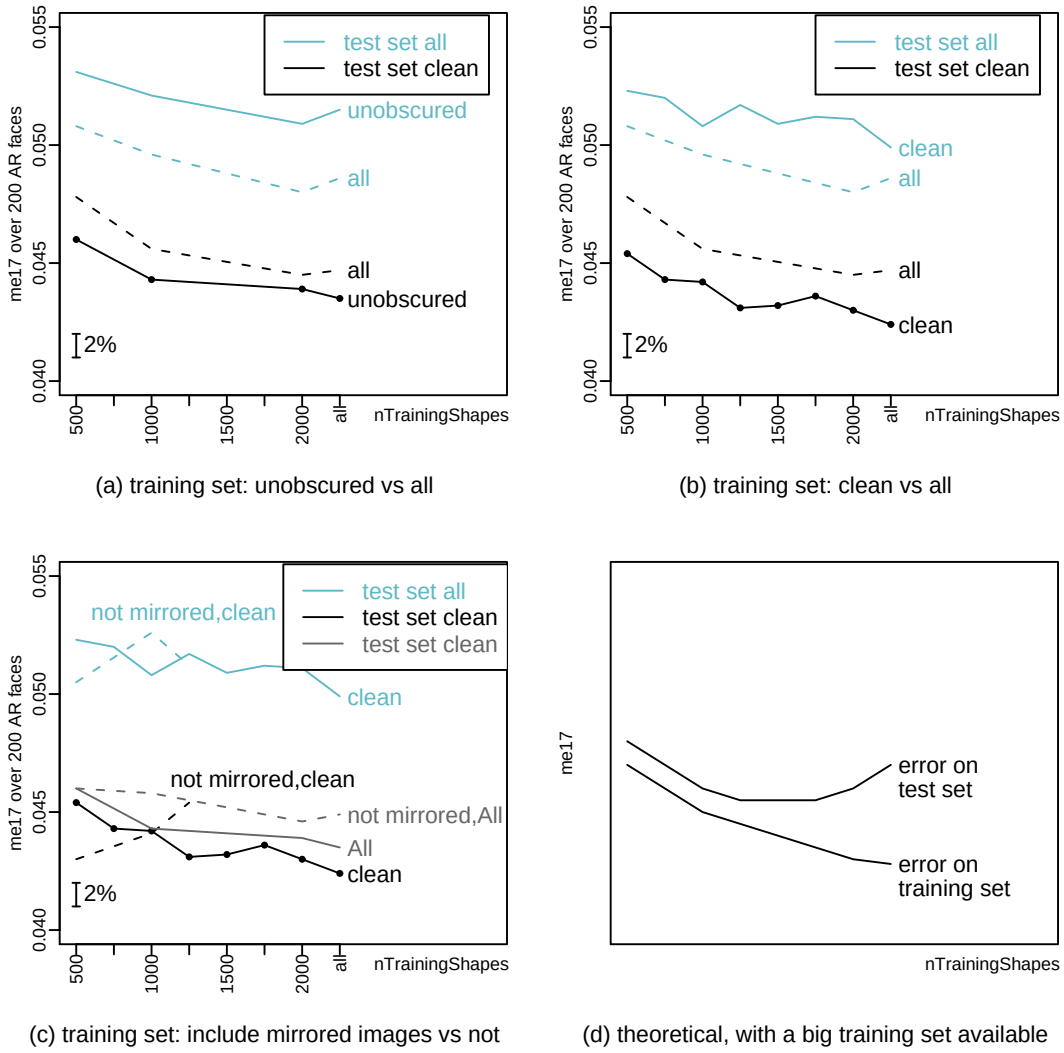


Figure 5.5: *Fit for various training and test sets, 1D profiles.*
Except where marked ***not mirrored****, the training sets include mirrored images.* ***Unobscured*** *means that the face is unobscured (by the edge of the image or by the subject's hand).* ***Clean*** *faces have no face attribute bits set (Table 4.2).*

5.4.9 Face attributes in the training set

Looking at the table of face attributes (Table 4.2) on page 26, it is natural to ask a few questions. For example, is it better to train on all images, or just on "good" images? And assuming we want to search for features only on faces without glasses, is it better to train on all faces or just on a subset? There are many possibilities. Figure 5.5 shows some of the more interesting ones.

The color of the curves indicates the test set. The annotation on the right of each curve is the training set. The horizontal axis is the number of training images. The **all** label means

all images in the specified training set — this size varies with the training set. The training sets include mirrored images except where marked **not mirrored** (section 4.1.2).

In the graphs, **unobscured** means that the face is unobscured i.e. the Obscured attribute bit is clear (Table 4.2). **Clean faces** have no face attribute bits set.

Using subfigure (b) as an example, we see that if one wishes to search for landmarks only on clean faces (black lines), then it is best to train the model only on clean faces. If one wishes to search for landmarks on on all images (blue lines), it is best to train on all images. Finally, the best results for clean faces (solid black) are better than the best results for all faces (dashed blue).

All graphs show a general tendency for the test error to decrease as the number of training images increases. That is true at least up to about 2000 training shapes; after that the error could increase or decrease. In the cases where error increases as the number of training images increases, we are getting a hint of the theoretical concave test curve shown in subfigure (d) (see for example Hastie et al. [40] Figure 2.11).

The main conclusions from the graphs are

1. as a general rule, the results depend more on the test set than the training set
2. for landmark location on a test set sampled from all AR images, the best XM2VTS training set is a set of 2000 images sampled from all XM2VTS images, including mirrored images (subfigure (a) dashed blue line, same as subfigure (b) dashed blue line). Note that this applies, as does everything in this section 5.4, only to models with 1D profiles.

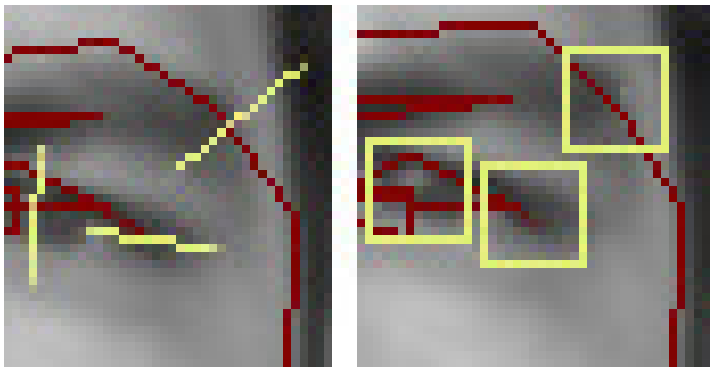


Figure 5.6: ***Left*** *some 1D profiles.* ***Right*** *some 2D profiles.*
Yellow shows the profiles (each 2D profile is the area inside a yellow box). Red shows the current shape model.

5.5 Models with 2D profiles

This section more-or-less repeats the entire previous section, but for ASMs that use two- instead of one-dimensional profiles.

5.5.1 About 2D profiles

The classical ASM uses a one-dimensional profile at each landmark. Now we are interested in using two-dimensional profiles. Instead of sampling a one-dimensional line of pixels along the whisker, we sample a square or circular region around the landmark (Figure 5.6). Intuitively, a 2D profile area captures more information around the landmark, and this information, if used wisely, should give better results.

During search we displace the sampling region in both the "x" and "y" directions, where x is orthogonal to the shape edge at the landmark and y is tangent to the shape edge.

In this project, 2D profiles stay aligned to the edges of the image. Unlike 1D profiles, we do not rotate the profiles to align with the shape model whisker (section 5.4.8). Without reorientation we must rely on the face being approximately upright, but do avoid interpolation noise.

The covariance matrix of a set of 2D profiles (section 2.4.2) is formed by treating each 2D profile matrix as a long vector (by appending the rows end to end), and calculating the covariance of the vectors.

We start off our investigation with profiles which are the sigmoid of the normalized gradient magnitude, with a flat window. These terms will be explained in section 5.8 where other profile types are considered.

File	Model	me17	Time
AR	1D	0.0	0.0
AR	2D	-4.4	272
XM2VTS	1D	0.0	0.0
XM2VTS	2D	-3.3	293

Table 5.11: *2D versus 1D profiles.*
Using 2D profiles give better results but is about 3 times slower.

5.5.2 2D versus 1D profiles

To see that we are not barking up the wrong tree, let's first do a quick check comparing 2D to 1D profiles. Table 5.11 shows that the me17s are better for 2D searches, but searches take much longer.

5.5.3 Face size scaling

As in section 5.3, we prescale faces to a common pixel width before using them. Table 5.12 shows the results. The width for the lowest AR error is not quite the same as that for 1D profiles (Table 5.2), but it is close, and to avoid keeping two image pyramids we continue using the 1D parameter values.

Model	AR	XM2VTS	
Reference	0.0	0.0	
width 140, nearest-pixel	1.3	0.7	
width 140, bilinear	-1.5	1.6	
width 160, nearest-pixel	3.3	-0.9	
width 160, bilinear	1.6	-0.2	
width 180, nearest-pixel	-0.9	-0.4	
width 180, bilinear	-3.0	-1.9	<chosen, see text
width 200, nearest-pixel	-4.0	-1.8	<lowest AR
width 200, bilinear	-2.4	-1.7	
width 220, nearest-pixel	-1.9	-0.5	
width 220, bilinear	-1.8	-1.8	
width 240, nearest-pixel	-0.9	-1.1	
width 240, bilinear	-0.3	-1.5	
width 250, nearest-pixel	1.8	-1.2	
width 250, bilinear	2.2	-1.8	

Table 5.12: *Scaled face width, 2D profiles.*
Here we prescale all faces to the given width before using them.

5.5.4 Profile model search parameters: nProfWidth and nPixSearch

During profile matching, we search in the vicinity of each landmark for the best profile match. Table 5.13 on the next page shows results for 2D profile models (compare to Table 5.3 for 1D profiles).

The fit can be improved beyond that for the chosen parameter values, but only at the expense of much more real time. This is a fairly arbitrary judgment call: balancing error versus time, knowing that in section 5.10 we will revisit profile widths.

The displacements on the table are "x" displacements (orthogonal to the shape model boundary). Not shown are the "y" displacements (tangent to the shape model boundary), but a y displacement of ± 1 gives good results.

```
Parameters are nProfWidth, nPixSearch

Model    AR XM2VTS  Time
Ref     0.0    0.0     0

9,1    -7.4    4.8   -58
9,2    -7.3    3.3   -49
9,3     6.5    5.7   -25
9,4    16.3    7.7     0
9,5    28.9   10.6    37
9,6    76.8    6.9    78
9,7   337.0    9.7   130

11,1  -10.1    3.9   -52
11,2  -10.9    0.6   -33  <chosen, because a low AR and good time
11,3    4.0    5.4     7
11,4   10.2    7.9    54
11,5   24.4    5.6   119
11,6   70.2    4.5   193
11,7  382.2    8.3   285

13,1  -10.7    3.4   -41
13,2  -11.2   -0.2    -5  <also considered but not chosen
13,3    1.9    3.4    60   because too slow
13,4    7.9    4.9   141
13,5   17.7    6.4   250
13,6   63.0    3.8   374
13,7  351.1    5.7   524

15,1  -11.7    4.1   -26
15,2  -12.7   -0.6    35  <lowest AR, this 1.8% better but
15,3   -0.3    3.6   139   much slower than the chosen params
15,4    5.9    4.8   270
15,5   15.8    7.2   442
15,6   60.6    1.7   643
15,7  415.7    3.5   881

17,1  -11.3    2.8    -4
17,2  -11.8   -0.6    93
17,3    1.2    3.4   251
17,4    7.8    5.3   455
17,5   15.2    5.9   716
17,6   57.1    0.4  1026

19,1  -10.8    1.9    27
19,2  -12.1   -0.9   179
19,3   -0.8    3.9   418
19,4    6.0    4.5   728
19,5   15.3    5.2  1126
```

Table 5.13: *nProfWidth and nPixSearch, 2D profiles.*

```
Parameters are nMaxSearchIters, nQualifyingDisplacements

Model   AR  XM2VTS  Time
2,75   3.1    0.3   -37
2,80   3.5   -0.5   -36
2,85   3.0   -0.2   -36
2,90   2.4   -0.8   -35
2,95   2.3   -0.6   -35

3,75   0.5   -0.2   -19
3,80  -0.3   -0.1   -17
3,85   0.0   -0.4   -15
3,90  -0.2   -0.2   -14
3,95  -0.7   -0.2   -13

4,75   0.0    0.0     0 <ref
4,80  -0.6   -0.2     2
4,85  -0.3   -0.5     4
4,90  -1.1   -0.9     8 <chosen, lowest AR, time ok
4,95  -0.6   -0.4     9

5,75   0.9   -0.3    15
5,80   0.0    0.3    20
5,85   0.3    0.6    24
5,90   0.9   -0.3    28
5,95   0.0    0.2    31

6,75   0.8   -0.2    32
6,80  -1.1    0.1    39 <also good AR, but slower
6,85   0.1    1.0    43
6,90   1.2   -0.2    50
6,95   0.6   -0.3    54

7,75   0.7    0.2    48
7,80  -0.5    0.1    56
7,85   0.1    0.2    63
7,90   0.6   -0.4    70
7,95   0.2   -0.7    75
```

Table 5.14: *nMaxSearchIters and nQualifyingDisplacements, 2D profiles.*

5.5.5 Search parameters: nMaxSearchIters and nQualifyingDisplacements

During profile matching, we search in the vicinity of each landmark for the best profile match. Table 5.14 shows results for 2D profile models (compare to Table 5.4 for 1D profiles). We see that good results can be obtained with `nMaxSearchIters=4` and `nQualifyingDisplacements=90`.

5.5.6 Shape model parameters: nEigs and bMax

After generating a suggested shape by profile matching, we conform the suggested shape to the shape model. We must choose the appropriate `nEigs` and `bMax` parameters for the shape model. The results for 2D models are quite similar to those for 1D models (section 5.4.3) and so we stay with the 1D model parameter values.

Model	AR	XM2VTS	Time	
Lev=0	-0.1	-1.1	22	
Lev=0 AND AllExceptMouth	0.0	0.0	0	<ref
Lev=0 AND Eyes	0.2	-0.2	-43	
Lev=0 AND Internal	-1.6	-1.9	2	
Lev=0 AND InternalExceptMouth	-1.1	-0.9	-17	
Lev=0 AND NotInternal	7.5	3.8	-37	
Lev<=1	0.8	-1.4	96	
Lev<=1 AND AllExceptMouth	-1.7	0.6	54	
Lev<=1 AND Eyes	-3.4	-3.0	-33	
Lev<=1 AND Internal	-1.7	-1.7	61	
Lev<=1 AND InternalExceptMouth	-4.1	-1.4	20	<chosen, lowest AR
AllLevs	5.2	-2.9	194	
AllLevs AllExceptMouth	3.1	0.6	124	
AllLevs AllInternalExceptMouth	-0.7	1.6	63	
AllLevs AllInternal	1.7	-4.4	130	
AllLevs AllNotInternal	14.9	7.6	4	
(Lev=0 AND AllExceptMouth) OR AnyLevEyes	0.1	-2.1	22	
(Lev<=1 AND AllExceptMouth) OR AnyLevEyes	1.3	-1.4	69	

Table 5.15: *Which profiles should be 2D?*

5.5.7 Which profiles should be 2D?

For best results not all profiles should be 2D; some should remain 1D. Table 5.15 shows a number of possibilities.

In the table, **Lev=0** means that 2D profiles are used only at pyramid level 0 (full resolution). **Lev<=1** means that 2D profiles are used at pyramid levels 1 and 0. **AllLevs** means that 2D profiles are used at all pyramid levels.

The table shows that best results are obtained using 2D profiles at pyramid levels 1 and 0 for the interior landmarks excluding the mouth landmarks. All other profiles remain 1D.

The table results can be a bit misleading. We are using the me17 to evaluate the results. The me17 uses only the interior landmarks, and thus favors interior landmarks, possibly at the expense of landmarks at the edge of the face.

Also of interest are the times — 2D profiles are much slower.

```
Parameters are nEigs and bMax for final pyramid level

Model      AR  XM2VTS

20,1.5    0.0    -1.1
20,1.8    0.0     0.0 <ref
20,2.0    0.0     0.2
20,2.6    0.4     1.3
20,3.0    1.0     1.3
20,5.0    1.2     1.1

24,1.5   -1.1    -0.8
24,1.8   -1.4    -0.2
24,2.0   -1.4     0.2
24,2.6   -0.7     1.0
24,3.0   -0.7     0.9
24,5.0   -0.3     1.0

26,1.5   -1.0    -1.1
26,1.8   -1.1    -0.3
26,2.0   -1.7    -0.1 <chosen, lowest AR
26,2.6   -1.0     0.7
26,3.0   -0.7     0.5
26,5.0   -0.1     0.6

30,1.5   -1.3    -0.6
30,1.8   -1.3    -0.2
30,2.0   -1.5     0.4
30,2.6   -1.0     1.1
30,3.0   -0.7     1.1
30,5.0   -0.2     0.8

40,1.5   -1.0    -0.3
40,1.8   -1.2     0.1
40,2.0   -0.8     0.7
40,2.6    0.0     1.3
40,3.0    0.1     1.2
40,5.0    0.0     1.0
```

Table 5.16: *Loosening up the shape model for the final iteration, 2D profiles.*

5.5.8 Loosening up the shape model for the final iteration

In section 5.4.4 we looked at a modification to the shape model: adding noise during training. Here we look at another modification. As usual, we conform the suggested shape to the shape model for each iteration of the search, at each pyramid level. But for the last pyramid level, we get improved results by using a higher value for `nEigs` and `bMax` (section 5.4.3). See Table 5.16.

This modification is effective only for 2D profiles, not for 1D profiles. This is probably because as the profile matches become more reliable the shape constraints can be weakened.

PyramidRatio	nPyramidLevs	Interp	AR	XM2VTS	Time	
2	4	bilinear	0.7	0.2	8	
sqrt2	4	bilinear	10.4	3.9	17	
sqrt2	6	bilinear	5.9	-1.7	46	
sqrt2	8	bilinear	2.7	-1.8	42	
sqrt3	4	bilinear	7.4	-0.2	14	
sqrt3	6	bilinear	1.5	1.8	21	
2	4	nearestPix	0.0	0.0	-2	<chosen
sqrt2	4	nearestPix	10.3	4.1	5	
sqrt2	6	nearestPix	6.6	-1.1	19	
sqrt2	8	nearestPix	-0.9	-1.2	24	
sqrt3	4	nearestPix	6.8	-0.4	5	
sqrt3	6	nearestPix	-0.8	-0.2	12	

Table 5.17: *Pyramid ratio, 2D profiles.*

5.5.9 Pyramid ratio and method of scaling images

We need to choose a pyramid scaling ratio and scaling method for 2D profiles, just as we did for 1D profiles (sections 5.4.6 and 5.4.7). See Table 5.17. The table shows that we can stay with the values used for 1D profiles: `PyramidRatio=2.0 nPyramidLevs=4`. Thus only one image pyramid is needed.

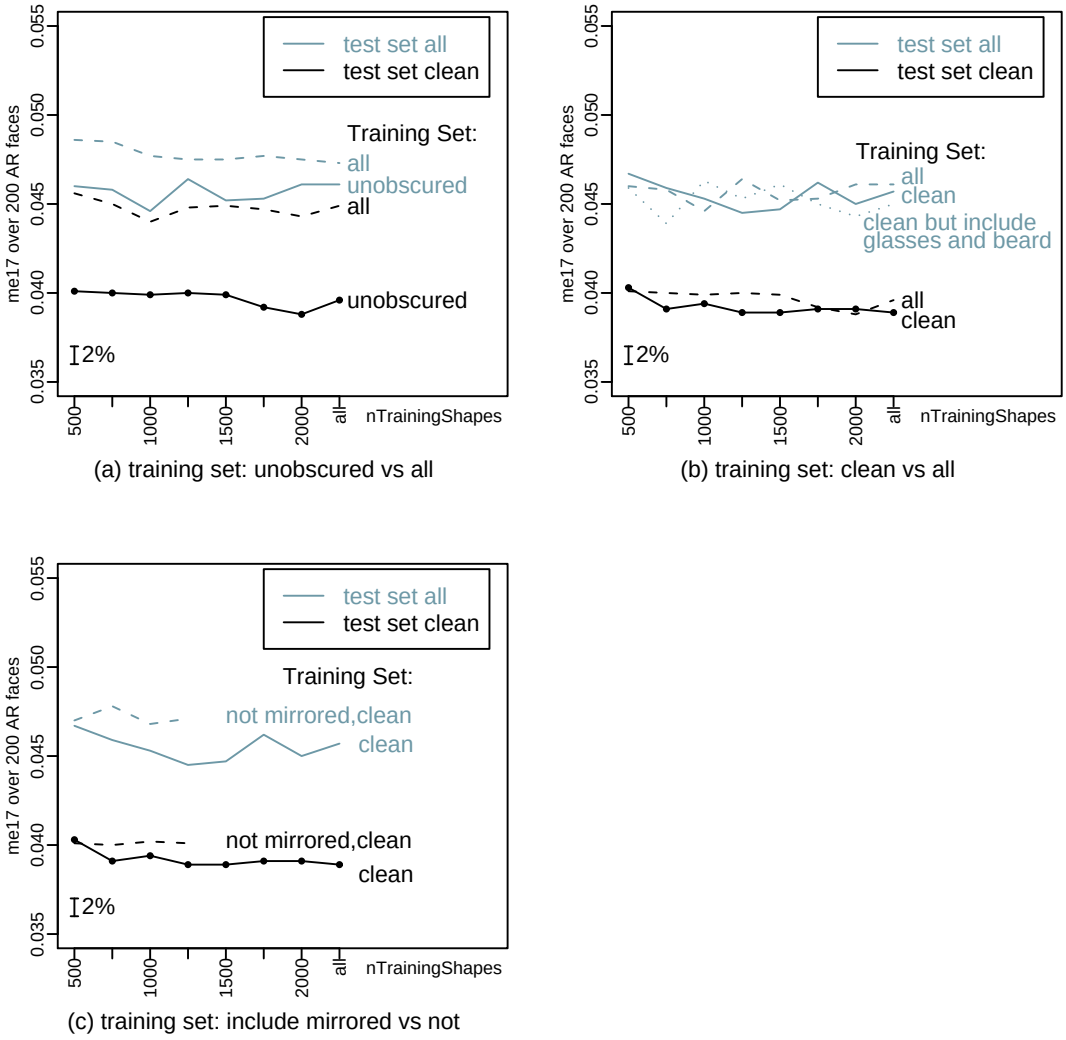


Figure 5.7: *Fit for various training and test sets, 2D profiles.*
Except where marked ***not mirrored****, the training sets include mirrored images.* ***Unobscured*** *means that the face is unobscured (by the edge of the image or by the subject's hand).* ***Clean*** *faces have no face attribute bits set (Table 4.2).*
In subfigure (b), the curve for `clean but include glasses and beard` *is not shown for* `test set clean` *because it is the same as the curve for* `clean`*.*

5.5.10 Face attributes in the training set

In section 5.4.9 we looked at how training and evaluating on images with different attributes affected the fit. Now we repeat that section, but for 2D profiles. See Figure 5.7.

Cutting through the details, we can summarize Figure 5.7 as follows: the training set that gives the best results on AR faces is the set of clean XM2VTS images, but including images with glasses and beards. The mirrored images should also be used. This is the dotted blue curve in subfigure (b).

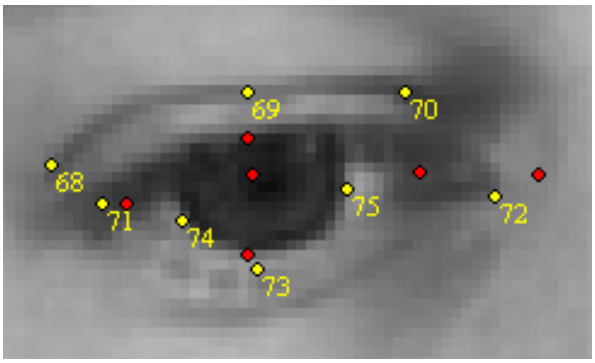


Figure 5.8: *Left eye with new landmarks in yellow, existing landmarks in red.*

		AR	XM2VTS	Time
Test set:	All faces			
	nLandmarks=68	0.0	0.0	0
	nLandmarks=76	-0.3	-1.4	13
	nLandmarks=84	-4.6	-4.9	23
Test set:	Clean faces			
	nLandmarks=68	0.0	0.0	0
	nLandmarks=76	1.1	-1.4	13
	nLandmarks=84	-3.2	-5.3	23

Table 5.18: *Expanded XM2VTS training set — overview.*
***Clean faces** have no face attribute bits set (Table 4.2).*

5.6 Adding landmarks to the XM2VTS set

In general, the more landmarks the better (section 5.1). Pursuing this improvement, eight landmarks were added to the left eye of clean faces in the XM2VTS set, as shown in Figure 5.8. (Left is with respect to the viewer.) The total number of landmarks becomes `68 + 8 = 76`.

Manually landmarking images is a tedious business, so instead of manually adding extra landmarks to the right eye, during training we generate 8 right eye landmarks by symmetry from the left eye. The total number of landmarks then becomes `68 + 8 + 8 = 84`.

There are limits to this process, even for an exceptionally patient human landmarker, because one tends to run out of logical positions for landmarks.

Table 5.18 shows that as we add landmarks, the me17s tend to improve.

Table 5.19 on the next page shows details. The improved fit at the eyes ripples across to cause improved fits on other facial features. Note that the left is much better than the right eye fit for the 76 point sets (which have new landmarks on the left eye only).

		Face	me17	Jaw	Nose	Mouth	LEye	REye
Test set: All faces								
Ref	nLandmarks=68	0.0	0.0	0.0	0.0	0.0	0.0	0.0
AR	nLandmarks=76	-0.1	-0.3	0.8	2.9	-0.3	-6.1	2.4
AR	nLandmarks=84	-4.0	-4.6	-0.4	-1.8	-1.9	-9.3	-14.9
XM2VTS	nLandmarks=76	-0.3	-1.4	1.9	-0.3	-2.5	-10.6	0.9
XM2VTS	nLandmarks=84	-2.5	-4.9	1.5	-4.9	-2.5	-12.6	-9.0
Test set: Clean faces								
Ref	nLandmarks=68	0.0	0.0	0.0	0.0	0.0	0.0	0.0
AR	nLandmarks=76	0.9	1.1	-0.8	6.9	0.2	-4.5	4.5
AR	nLandmarks=84	-2.8	-3.2	-0.6	0.0	0.1	-8.8	-9.9
XM2VTS	nLandmarks=76	-0.5	-1.4	0.7	-0.6	-0.1	-8.1	0.9
XM2VTS	nLandmarks=84	-2.3	-5.3	1.5	-6.0	-1.4	-10.5	-13.1

Table 5.19: *Expanded XM2VTS training set — details.*

5.7 Stacking models

Accurate positioning of the start shape is crucial (section 5.2). In this section we look at a method of better positioning the start shape, by stacking two ASM models as follows:

1. run the ASM search on the image
2. run the ASM search again, using the results of the first search as the start shape.

Table 5.20 below shows some results. In the table, the `single model` is the best 2D model we have. The `stacked model` is the best 1D model we have followed by the best 2D model we have.

For `all faces` there is a large improvement in fit. For `clean faces` the improvement is small, because the start shape is already good for most of the clean faces.

Tests (not shown here) show that for stacked models it is fine to use a 1D model for the first search. Using a 2D model does not give better results, and takes longer. Other tests show that is suffices to start the second model at pyramid level 1, one level below full size.

	AR	XM2VTS
Test set: all faces		
single model	0.0	0.0
stacked model	-6.1	-14.0
Test set: clean faces		
single model	0.0	0.0
stacked model	-1.8	-0.1

Table 5.20: *Stacked model versus single model.*

0 0 0 0 1 0 0 0 0	0 1 0 1 -4 1 0 1 0	1 1 1 1 -8 1 1 1 1
mask_gray (gray level)	mask_4_lap (4 connected Laplacian)	mask_8_lap (8 connected Laplacian)
0 0 0 0 -2 1 0 1 0	0 0 0 0 -1 1 0 0 0	0 0 0 0 -1 0 0 1 0
mask_grad (gradient)	mask_x_grad (x gradient)	mask_y_grad (y gradient)

Table 5.21: *2D convolution masks.*

5.8 Profile types

A 2D profile at a landmark is created in four steps: convolution, normalization, equalization, and weight masking. These four steps are repeated at each landmark. This section describes each of the steps and then evaluates different profile types.

5.8.1 Convolution

In the convolution step, we move a 3x3 convolution mask over each pixel in the area around the landmark to generate a *profile matrix*. Each element of the profile matrix corresponds to a pixel. The element's value is a measure of some image property at the pixel. The property in question is determined by the convolution mask. This is a standard technique described in most image processing textbooks, for example Gonzalez and Woods [37].

Table 5.21 shows convolution masks used in this project. Taking `mask_x_grad` as an example, each element in the profile matrix is minus the value of the center pixel plus the value of the pixel to its right. Thus `mask_x_grad` gives the intensity gradient in the horizontal direction.

Experiments (not shown here) show that the best me17 results are obtained using these 3x3 masks. Other masks, or larger masks, do not give better results.

5.8.2 Beyond linear convolution

In this section we discuss a few nonlinear transformations that go beyond simple convolution.

1. **Mask_grad_mag**: This is the method we have been using in this chapter up to now. These profiles are similar to `mask_grad` but use the *magnitude* of the gradient as follows: (a) Take the difference between the pixel at the landmark and the pixel to its right.

(b) Take the difference between the pixel at the landmark and the pixel below it.
(c) Square the two differences, sum them, and use the square root of the sum.

2. Harris and Stephens [38] present an ingenious edge and corner detector. They calculate the difference between the image and itself as one is scanned over the other. They apply a gaussian window to this difference around the landmark. They then take a first order approximation, resulting in a quadratic which gives the intensity rate of change at each pixel. The two eigenvalues of the coefficents of the quadratic give a measure of cornerness (both eigenvalues big) or edgeness (one eigenvalue big). Since the eigenvalues measure the principal rates of change they are invariant to image rotation.

   In this project, simple templating was found to give better results than the Harris Stephens method.

3. Scott et al. [60] extend the Harris Stephens detector to yield two numbers at each pixel. The one number is a measure of edgeness at the pixel; the other is an independent measure of cornerness.

   In this project, this technique was found to perform poorly. However, Scott et al. and other researchers [23] report good results (albeit in an Active Appearance Model context). Further investigation is needed.

5.8.3 Normalization

Convolution yields a profile matrix for the landmark. We normalize the profile matrix by dividing it either (a) by its **length** or (b) by the **absolute sum** of its elements. The length here is the Frobenius norm i.e. the euclidean norm if you treat the matrix as a single long vector.

Normalization corrects for varying overall image brightnesses and contrast.

5.8.4 Equalization

We equalize the normalized profile matrix by applying a sigmoid transform to each element x of the matrix:

$$x' = \frac{x}{abs(x) + c} \ . \tag{5.1}$$

The *shape constant* c determines the shape of the sigmoid. Figure 5.9 on the next page shows some sigmoids with different values of c. Sigmoids compress big positive or negative values but are approximately linear for input values near 0. Using this transformation on the image intensity reduces the effect of outliers [19]. Smaller values of c make the compression more pronounced.

In this project, the best values for sigmoid constant were found to be fairly high (greater than about 10) i.e. just a small amount of sigmoid equalization is best.

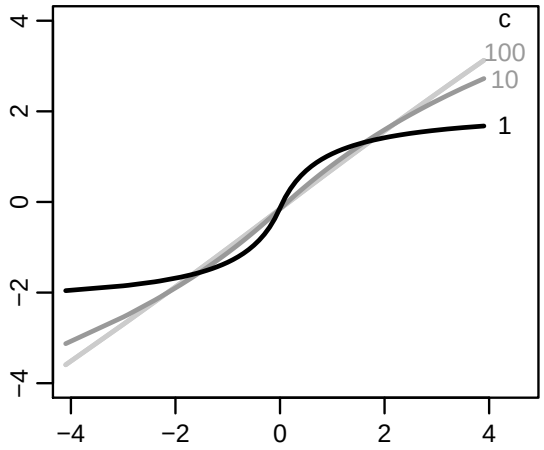


Figure 5.9: *Sigmoids for different values of the shape constant c.*

5.8.5 Weight masking

Finally, we weight the values in the profile matrix to give more weight to pixels near the landmark. Figure 5.10 below shows the three weight masks used in this project. Elements outside the range of the circular mask are set to zero. The gaussian mask is formed by

$$x' = x \; exp\left(\frac{-(i^2 + j^2)}{2\sigma^2}\right) \tag{5.2}$$

where σ^2 is a positive constant determining the steepness of the gaussian peak, and i and j are the horizontal and vertical distances in pixels from the landmark.

In this project, weight masks were found to give small or no improvement in overall fit, although only a couple of different values of σ^2 were tried.

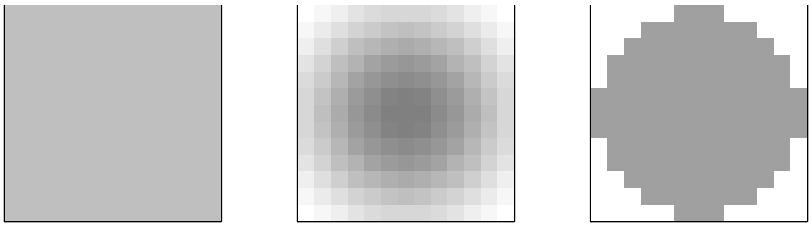


Figure 5.10: *Weight masks.*

Convolution	Norm	Equalization	AR	XM2VTS	
mask_grad_mag	abs_sum	sigmoid_shape10	5.2	-0.8	"shape10" means shape const c = 10
		sigmoid_shape100	17.2	5.7	"shape100" means shape const c = 100
	len	sigmoid_shape10	0.0	0.0	<ref
		sigmoid_shape100	0.5	-0.2	
		none	0.1	-0.2	
mask_grad	abs_sum	sigmoid_shape10	-3.8	-2.3	<tied lowest AR
		sigmoid_shape100	-3.8	-2.9	<chosen, tied lowest AR
	len	sigmoid_shape10	-1.6	-0.6	
		sigmoid_shape100	-1.8	-4.0	
		none	-2.0	-1.7	
mask_y_grad	abs_sum	sigmoid_shape10	0.3	-0.3	
		sigmoid_shape100	6.0	1.9	
	len	sigmoid_shape10	1.8	-0.5	
		sigmoid_shape100	6.4	1.2	
		none	7.0	1.1	
mask_4_lap	abs_sum	sigmoid_shape10	-2.3	-2.6	
		sigmoid_shape100	-2.2	-2.8	
	len	sigmoid_shape10	-0.1	-0.8	
		sigmoid_shape100	0.2	-1.5	
		none	0.5	-1.2	
mask_8_lap	abs_sum	sigmoid_shape10	0.2	-1.4	
		sigmoid_shape100	0.3	-0.3	
	len	sigmoid_shape10	-0.1	1.1	
		sigmoid_shape100	0.7	0.9	
		none	0.6	0.9	
harris_steph	abs_sum	sigmoid_shape10	7.5	9.9	
	len	sigmoid_shape10	11.6	16.5	

Table 5.22: *2D profile types.The weight mask is flat for all the above tests.*

5.8.6 Measurements

Table 5.22 shows some of the more interesting results for different profile types.

The reference is the model we have been using in this chapter up to now

`mask_grad_mag len sigmoid_shape10`.

However, the table shows that results are best using

`mask_grad abs_sum sigmoid_shape100`

so we will use that for the rest of this dissertation.

Profile1		Profile2		AR		XM2VTS			
Convolution	Norm	Convolution	Norm	mean	median	mean	median	time	
mask_grad	abs_sum	---- none --	---	0.0	0.0	0.0	0.0	0	<ref
mask_x_grad	abs_sum	mask_y_grad	abs_sum	-2.6	-7.9	0.3	1.7	54	
mask_x_grad	abs_sum	mask_y_grad	len	-2.1	-8.5	0.8	-0.1	53	
mask_4_lap	abs_sum	mask_grad	abs_sum	-1.1	-0.3	-0.7	1.9	52	
mask_x_grad	abs_sum	mask_grad	abs_sum	-0.6	-4.0	2.9	1.7	52	
mask_4_lap	abs_sum	mask_y_grad	len	-0.4	-7.2	1.1	2.9	53	
mask_4_lap	abs_sum	mask_y_grad	abs_sum	-0.3	-5.2	0.5	1.2	53	
mask_grad	len	mask_4_lap	abs_sum	-0.3	-2.8	0.4	5.0	52	
mask_grad	abs_sum	mask_8_lap	abs_sum	-0.2	-1.7	-0.3	4.4	52	
mask_grad	abs_sum	mask_4_lap	len	-0.1	-3.7	-0.1	0.1	52	
mask_y_grad	abs_sum	mask_8_lap	abs_sum	0.1	-6.3	0.4	0.9	54	
mask_y_grad	abs_sum	mask_4_lap	len	0.1	-6.2	1.1	3.0	57	
mask_grad	abs_sum	mask_8_lap	len	0.2	-1.8	0.5	2.5	52	
mask_grad	len	mask_8_lap	abs_sum	0.6	-2.9	0.4	1.9	52	
mask_grad	len	mask_y_grad	abs_sum	0.9	-4.9	0.9	2.0	52	
mask_grad	len	mask_x_grad	abs_sum	0.9	-1.1	3.9	8.1	52	
mask_grad	len	mask_grad	abs_sum	1.0	-3.6	0.7	0.6	51	
mask_grad	abs_sum	mask_y_grad	abs_sum	1.0	-3.7	-0.1	3.9	52	
mask_grad	len	mask_4_lap	len	1.0	-7.2	0.6	2.9	53	
mask_y_grad	abs_sum	mask_8_lap	len	1.1	-6.2	0.7	1.5	54	
mask_grad	len	mask_8_lap	len	1.4	-3.0	1.5	2.3	52	
mask_grad	len	mask_y_grad	len	1.4	-5.5	1.0	2.1	56	
mask_grad	abs_sum	mask_y_grad	len	1.5	-4.1	0.6	-0.2	56	
mask_y_grad	abs_sum	harris_ste	abs_sum	1.5	-7.4	1.4	2.2	81	

Table 5.23: *Dual profiles, sorted on AR mean.*
All models use ***sigmoid shape100*** *equalization.*
Blank lines have been inserted to enhance readability and have no special meaning.

5.9 Dual profiles

In this section we evaluate the following technique:

1. Perform two independent profile matching searches around the landmark, using two different convolution masks. This will give two suggested new positions for the landmark.
2. Take the (equally weighted) mean of these two positions as the final suggested landmark position.

Table 5.23 shows results for dual profile models, sorted on the AR mean me17. The table shows just the best results, but other profile types were tried. The reference model is the best model from the previous section: `mask_grad abs_sum sigmoid_shape100`.

For the AR validation set, we get better medians but the means may or may not be better. A better median but worse mean usually indicates that most of the fits are better but the worst cases are worse.

For the XM2VTS training set, the results are on the whole not as good as using a single profile model.

Since these results are equivocal, we will not use dual profiles but rather will continue using the single profile approach described in section 5.8.

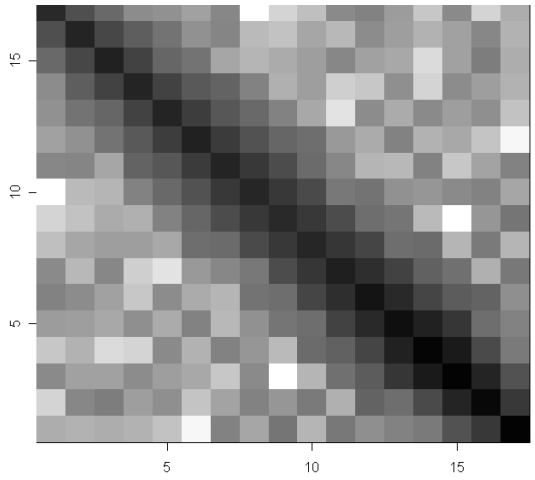


Figure 5.11: *Profile covariance matrix for a 1D profile (landmark 7, tip of chin)*
Bigger absolute values are darker.
The figure shows the log of the absolute values in the profile covariance matrix.

5.10 Trimming the profile covariance matrix

In this section we discuss trimming the profile covariance matrices. The resulting model is the *TrimmedModel*, which is the best model in this dissertation. The TrimmedModel also incorporates the improvements presented earlier in this chapter.

5.10.1 Trimming

The covariance between two pixels in a profile tends to be much higher for pixels that are closer together. This means that for 1D profiles, as we move away from the diagonal of the covariance matrix, the absolute values of the elements get smaller. You can see this in Figure 5.11, which shows the log of the absolute values. Intuitively then, we can simply ignore elements that are too far from the diagonal, or equivalently set them to 0. This is called *trimming* the covariance matrix.

For 2D profiles the distance between pixels is not simply the distance from the diagonal. Remember that covariance matrices for 2D profiles are formed by treating the 2D profile as one long vector, appending the rows end to end. Such covariance matrices can be trimmed with a little code to calculate the inter-pixel distances.

5.10.2 Why we trim

The advantage of trimming is faster landmark searches. The profile search time at each landmark is dominated by the calculation of the Mahalanobis distance in equation 2.6, reproduced

here:

$$Distance = (\boldsymbol{g} - \bar{\boldsymbol{g}})^T \boldsymbol{S}_g^{-1} (\boldsymbol{g} - \bar{\boldsymbol{g}}) \ . \tag{5.3}$$

Representing the trimmed matrix as a sparse matrix, because most entries are zero, means that we can calculate the Mahalanobis distance quickly — fewer additions and multiplications are needed.

We only trim 2D profiles. Trimming 1D profiles results in a neglible overall speed increase. The size difference between the original and trimmed covariance matrices is not big enough.

Some numbers for 1D profiles: a covariance matrix for a 13 pixel long 1D profile has 13 * 13 = 169 elements. If we clear elements for pixels that are more than 3 pixels apart, about (3 + 1 + 3) * 13 = 91 = 54% of the original entries remain. And for 2D profiles: a covariance matrix for a 13x13 2D profile has 169 * 169 = 28,561 elements. If we clear elements for pixels that are more than 3 pixels apart, only 6% of the original entries remain. See the source file `tcovar.cpp` for details.

The main reason we trim is for speed, but trimming also finds justification in the fact that it does not make statistical sense to estimate over 28,000 covariance values from approximately 1000 images in the training set.

5.10.3 Forcing positive definiteness

But there is a problem. Trimming results in a matrix that may no longer be positive definite. The Mahalanobis distance is only meaningful if the covariance matrix is positive definite. Or, equivalently, the distance should never be negative, and zero only if there is a perfect profile match. The usual meaning of distance breaks down if distances can be negative — for example, the shortest distance between two points may not be a straight line. (See, for example, Duda et al. [31] section 4.6 which discusses distance as a *metric*.)

The solution is to force the trimmed covariance matrix to be positive definite. One way of doing that is:

1. Trim the covariance matrix A by clearing entries for pixels that are not near each other.
2. Perform a spectral decomposition $A = Q \Lambda Q^T$ where Λ is a diagonal matrix of eigenvalues.
3. Set small or negative elements of Λ to a small positive value to create Λ'. The "small positive value" used in this project is $10^{-3} \times trace(A)/nrows(A)$.
4. Reconstruct A using Λ': $A = Q \Lambda' Q^T$.
5. Re-trim the new A by clearing entries for pixels that are not near each other.

A couple of iterations of the above algorithm does the trick: the final matrix A will be both trimmed and positive definite.

More rigorous ways of forcing positive definiteness are discussed in Gentle [35].

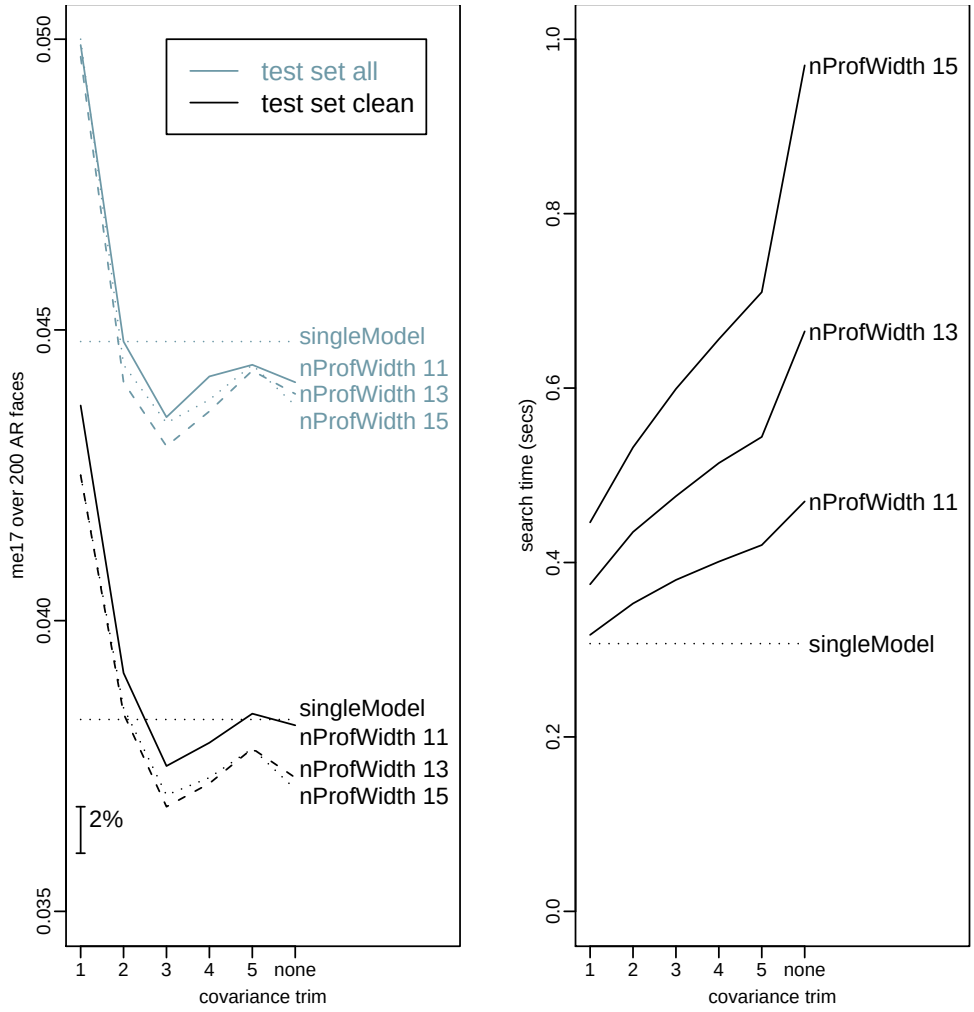


Figure 5.12: *Profile width and covariance trim.*
The parameters are for the second model of a stacked model.
***Left** AR fit*
***Right** search time*

5.10.4 Parameters: nProfWidth and nTrim

We need to determine two parameters for trimmed covariance matrices:

1. `nProfWidth`, the profile width in pixels.
2. `nTrim`. Entries for pixels that are more than `nTrim` pixels apart are cleared.

Actually, we have already determined the tentative best value of `nProfWith=11` in section 5.5.4. Now we redo the test using trimmed matrices, as shown in Figure 5.12. We obtain good me17s and times with `nProfWidth=13` and `nTrim=3`.

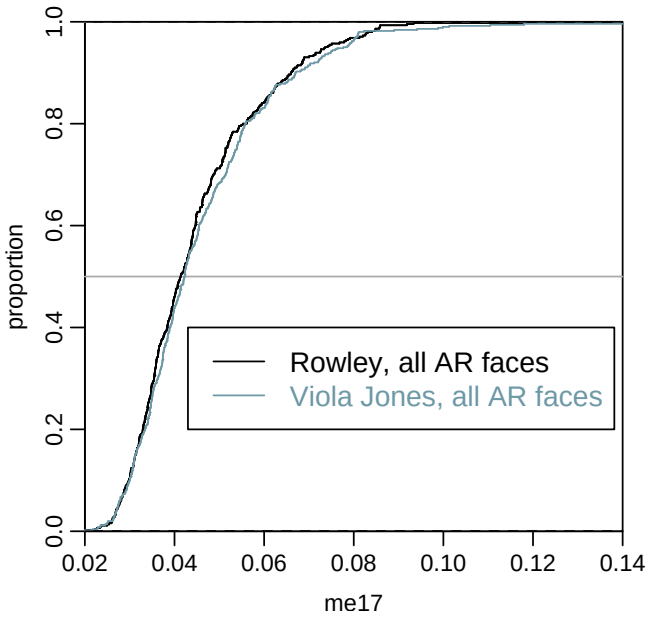


Figure 5.13: *Start shape from Rowley versus Viola Jones detector.*

5.11 Using the Viola Jones face detector

Up to now we have been using the Rowley face detector to determine the global position of the face, and hence to determine the start shape. In this section we replace it with the Viola Jones detector (section 4.4).

The start shape is generated as follows from the Viola Jones face rectangle. The width of the face rectangle is multiplied by a constant WidthCorrect. The rectangle is moved upwards by a constant HeightCorrect times its height. The model mean shape is aligned to the modified rectangle with a similarity transform, and becomes the start shape.

Table 5.25 on the next page shows a test done to estimate the values of `WidthCorrect=1.16` and `HeightCorrect=0.64`.

Figure 5.13 above and Table 5.24 below show the fits after generating the start shape with the Viola Jones detector. On the AR validation set the results are quite close to the Rowley results, but not so on the XM2VTS training set.

	AR	XM2VTS
Rowley	0.0	0.0
Viola Jones	4.4	16.4

Table 5.24: *Start shape from Rowley versus Viola Jones detector.*

Parameters are WidthCorrect, HeightCorrect

	AR	XM2VTS	
Reference	0.0	0.0	
1.08, 0.60	12.0	-1.6	
1.12, 0.60	1.4	-8.3	
1.16, 0.60	11.1	-10.3	
1.20, 0.60	13.7	-10.0	
1.24, 0.60	16.5	-7.1	
1.28, 0.60	19.2	-7.4	
1.08, 0.62	4.4	-4.9	
1.12, 0.62	1.3	-7.6	
1.16, 0.62	3.7	-10.1	
1.20, 0.62	7.0	-12.2	
1.24, 0.62	7.9	-8.6	
1.28, 0.62	19.0	-10.5	
1.08, 0.64	0.5	-7.8	
1.12, 0.64	0.3	-9.2	
1.16, 0.64	-1.3	-9.8	<chosen, lowest AR
1.20, 0.64	4.1	-10.4	
1.24, 0.64	8.7	-9.2	
1.28, 0.64	14.9	-9.9	
1.08, 0.66	-0.4	-3.9	
1.12, 0.66	-1.0	-5.2	
1.16, 0.66	-1.1	-11.5	
1.20, 0.66	0.3	-9.5	
1.24, 0.66	2.6	-3.6	
1.28, 0.66	5.6	-10.0	

Table 5.25: *Parameters for the Viola Jones face detector.*

Chapter 6

Results

In this chapter we will evaluate the performance of the TrimmedModel, which is the best model in this dissertation (section 5.10).

In a nutshell, the TrimmedModel stacks two models in a row, the second being an 84 point model with some 2D profiles and trimmed covariance matrices. We will use the Rowley face detector to determine the start shape, except where stated otherwise.

For the first time, we examine results on the BioId test set. As usual, the results are presented in terms of the me17 measure (section 4.5).

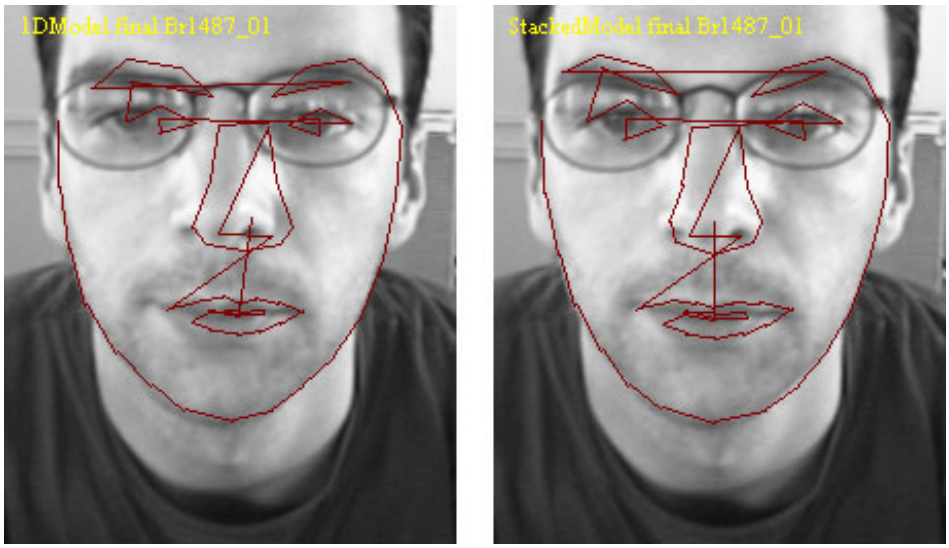


Figure 6.1: *Optimized classical ASM (left) versus TrimmedModel (right).*
The most visible improvement in the right image is in the position of the nose landmarks, but other improvements can also be seen.
The landmarks in the right image are still not faultless. Look at the lines around the base of the left side of the nose, for example.

6.1 Results on an example face

To start off, let's look at the improved results on just a single BioId face.

Figure 6.1 shows search results using a classical ASM and then using the TrimmedModel. The image is the same as Figure 1.2. The classical ASM used for this example is an optimized 68 point model with 1D profiles. The me17 has gone from 0.11 to 0.06, with visible improvements in landmark positioning.

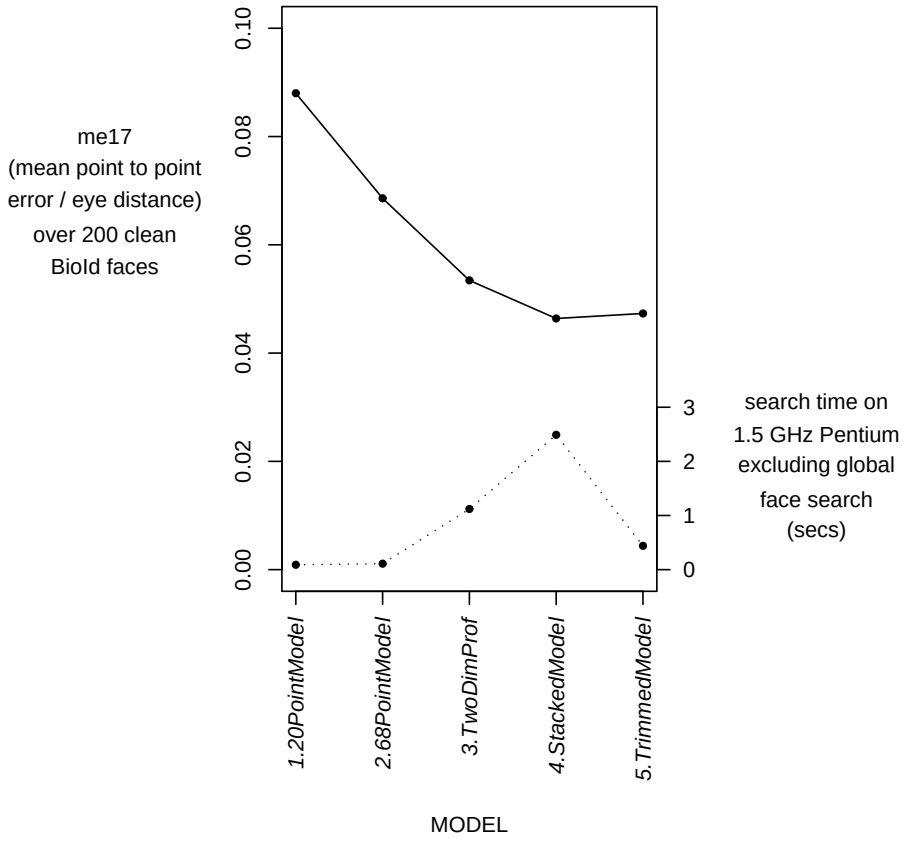


Figure 6.2: *Performance of various models.*

6.2 Relative performance of various models

In the last chapter it was easy to get lost in the details of model building. This section gives an overview by summarizing the performance of various models, as shown in Figure 6.2. This figure was created in reverse, by starting with the TrimmedModel and removing features one by one.

Each graph point represents the me17 averaged over all faces in the test set, for the given model. The test set consisted of 200 images randomly chosen from the clean BioId faces. *Clean* faces have no face attribute bits set (Table 4.2). A different BioId subset would give different results but would not make large changes to the *relative* performance of the various models.

The models shown in the graph are:

1. **20pointModel** The classical ASM built with 1D profiles and 20 landmarks — train on the XM2VTS set but use only those 20 XM2VTS landmarks in common with the BioId landmarks.

2. **68pointModel** As above, but use 68 XM2VTS landmarks (section 5.1). During search,

fit all 68 landmark (but calculate the me17, as always, from only 17 of landmarks).

3. **TwoDimProf** As above but:
   (a) Use 2D profiles for certain landmarks (section 5.5.1). The profiles are the sigmoid of the normalized gradient magnitude, with a 13x13 square window (section 5.8).
   (b) Generate the start shape using the position of the eyes located by the Rowley global face detector, as well as the overall position of the face (section 5.2).
   (c) When building the shape model, add noise to each landmark and also randomly stretch and contract the right hand side of faces (section 5.4.4).
   (d) Use a larger lambda and a larger number of eigenvectors in the final fit of the shape model (section 5.5.8).
4. **StackedModel** As above but:
   (a) Add 16 extra eye landmarks to a subset of the XM2VTS data (section 5.6). This is done by manually adding 8 landmarks to the left eye and, by symmetry, generating 8 landmarks for the right eye. Build a two-dimensional ASM by training as before, but on the extended XM2VTS data.
   (b) During search, run two models in series as follows (i) run the classical 1D profile ASM (ii) run the 2D ASM, using the results of the first search as the start shape (section 5.7).
5. **TrimmedModel** As above, but ignore the covariance of pixels that are more than 3 pixels apart (section 5.10). This improves speed without unduly affecting the fit.

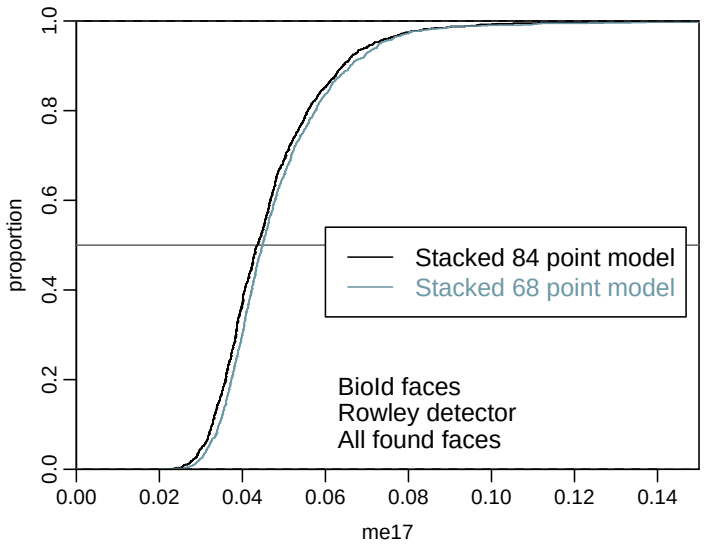


Figure 6.3: *84 versus 68 point model.*

	Face	me17	Jaw	Nose	Mouth	Eyes
TrimmedModel	0.0	0.0	0.0	0.0	0.0	0.0
TrimmedModel-68	3.1	3.4	0.0	1.5	0.5	13.9

Table 6.1: *84 versus 68 point model.*
TrimmedModel *is our best 84 point model.*
TrimmedModel-68 *is identical to TrimmedModel, except that the model is built with the original 68 point XM2VTS set.*
The test set is all BioId faces found by the Rowley detector.

6.3 84 versus 68 point model

In section 5.6 we extended the XM2VTS training database by adding 16 landmarks. Now we ask the question: what is the best ASM possible using the techniques in this thesis, but with the original unmodified XM2VTS dataset?

Figure 6.3 and Table 6.1 show that training on the original set gives worse me17 results on the BioId data, but the deterioration is small.

Note the poor results for the eyes. This is not surprising, because the additional landmarks on the 84 point model were added around the eyes.

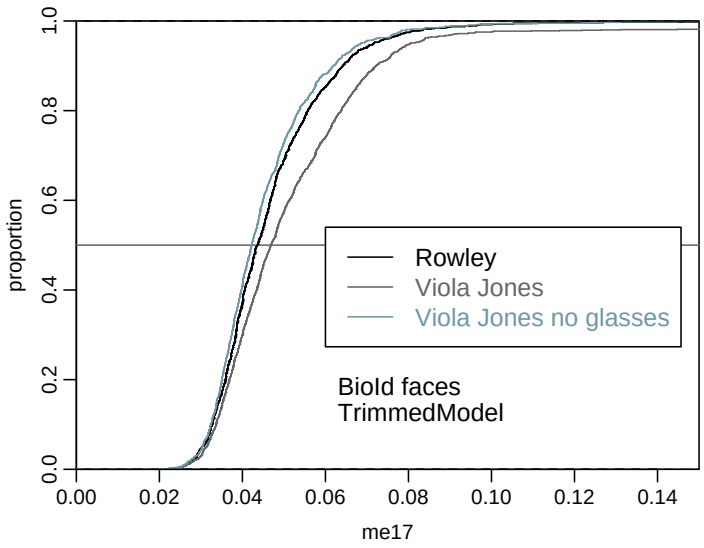


Figure 6.4: *Start shape from Rowley versus Viola Jones detector*

6.4 Rowley versus Viola Jones

We generate the start shape for a search using a global face detector. How does the Rowley stack up against the Viola Jones detector in this application?

Figure 6.4 appears to show that the Rowley detector gives better results. There is a misleading subtlety though: the Rowley detector cannot find "hard" faces (Table 4.1), Thus the black curve excludes hard faces, giving the Rowley detector an unfair advantage. Also, the Rowley detector is much slower (Table 4.1). Thus, depending on the application, it may be preferable to use the Viola Jones detector.

In detail, the curves in the graph are:

1. **Rowley** is the TrimmedModel with the Rowley detector, applied to the BioId faces (1144 faces found by the detector).
   From the figure, the me17s at the 50 and 90 percentiles are roughly 0.04 and 0.06. Examples of faces with these values can be seen in Figure 4.3.

2. **Viola Jones** is the TrimmedModel with the Viola Jones detector, applied to the BioId faces (1456 faces found by the detector).

3. **Viola Jones no glasses** is the TrimmedModel with the Viola Jones detector, applied to BioId faces without glasses (1031 faces).

From the graph we can see that the worst case performance for `Viola Jones all` is poor. The gray curve near the top of the graph flattens out at about the 96th percentile. It turns out that the remaining 4% faces are all wearing glasses, and the Viola Jones detector overestimates

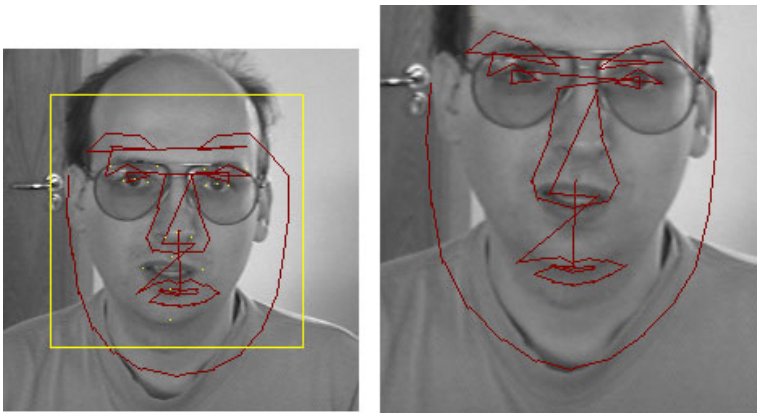


Figure 6.5: *Effect of a bad start shape from the face detector.*
Left *The yellow box is the face region detected by the Viola Jones face detector The red shape is the start shape i.e. the model mean shape aligned to the Viola Jones box. Because the detected face region is too big, the start shape is too big.*
Right *Bad ASM search results because of the bad start shape.*

the size of these faces, as shown in Figure 6.5. A remedy is to exclude faces wearing glasses, giving the blue curve.

It is interesting to compare Figure 6.4 to Figure 5.13 i.e. to compare tests on the BioId set to tests on the AR set, both with the Viola Jones detector. The worst case performance on the AR set is better, probably because the model parameters were optimized in Chapter 5 for the AR set.

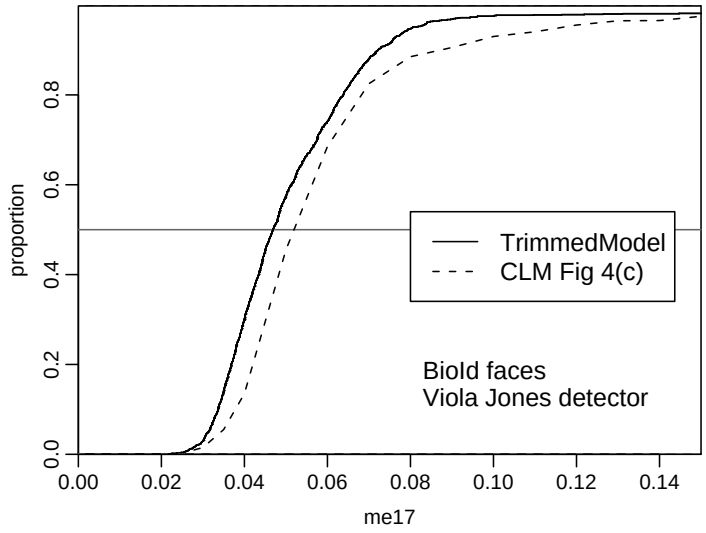


Figure 6.6: *Comparison to published results, with the Viola Jones detector.*

6.5 Comparison to published results

A fairly meaningful comparison can be made against the *Constrained Local Model* described in Cristinacce [28]. The results in Cristinacce's paper appear to be the best previously published results on the BioId set, and are presented in terms of the me17, which makes a direct comparison possible.

Briefly, the Constrained Local Model is similar to an Active Appearance Model [13], but instead of modeling texture across the whole face it models a set of local feature templates. During search, the feature templates are matched to the image using an efficient shape constrained search. The model is more accurate and more robust than the original Active Appearance Model.

The comparison here is unfair to Cristinacce, because his aims were different from those of this project. He was comparing the relative performance of novel algorithms, and to do so kept the framework in which these algorithms operate constant. But in this project, we optimize by changing the framework — by increasing the number of landmarks, for example. Nevertheless some comparison is needed, and Figure 6.6 shows the results.

The Viola Jones detector was used to find the faces (and generate the start shape) for both curves in the graph. **TrimmedModel** is the best model in this dissertation (section 5.10) applied to the BioId data. **CLM Fig 4(c)** is Cristinacce's Constrained Local Model applied to the same data. This curve reproduces the CLM curve in [28] Figure 4(c).

6.5.1 Search times

The TrimmedModel is just a little slower than the Constrained Local Model. The TrimmedModel mean search time is 440 ms. This time is for BioId images on a 1.5 GHz Pentium, and excludes the global face search time (Figure 6.2).

The equivalent time for a Constrained Local Model search is roughly 410 ms. This time is derived from section 4.4 in [28] as follows

```
240 * 2 - 70 = 410 ms.
```

The `*2` scales a 3 to a 1.5 GHz processor. The `-70` excludes the Viola Jones search time, but this is to a certain extent a guess based on the last line of Table 4.1.

6.6 Back in the real world

Testing against standard face databases is one thing, but how does the ASM do on "real" images? Figures 6.7 and 6.8 overleaf show the results of an informal test.

These figures were created as follows. From my personal collection of snapshots, eighteen images were selected of unobstructed more-or-less frontal views of upright faces with both eyes open. It turns out that such images are rare — nearly all faces in the collection are tilted or not facing the camera directly. An additional qualification was that the face and both eyes could be found by the Rowley detector. The backgrounds and face sizes were diverse. The TrimmedModel was then run on the images, and the images cropped to the faces for presentation.

The TrimmedModel has trouble with faces not facing the camera directly (for example, the jaw line in faces 13 and 14). It also has trouble with faces showing expression (for example, the mouth landmarks in faces 9 and 12). Another common problem is that the TrimmedModel confuses the chin for the shirt collar, especially if the start face is too big (Figure 6.5). A little blurring seems just a minor hindrance (for example, faces 2, 3, and 4).

The last three faces are drawings and paintings. Surprisingly, the results are not too bad on faces 17 and 18.

Although the TrimmedModel has had certain success, a human landmarker would do better on most of these images.

6.7 Further tests

Lie Gu provides a web interface to the detector of Zhou et al. [70]. The web interface returns a JPEG image with the feature shape superimposed on the face. It does not return printed landmark coordinates, which makes it hard to compare results. Visually, on the eighteen faces in figures 6.8 and 6.8, the detector seems to be sometimes better and sometimes worse than the TrimmedModel.

It would be useful to test against more facial feature detectors. Unfortunately, there appear to be no other publicly available detector software pre-configured for finding features in faces.

Figure 6.7: *Landmarks generated by the TrimmedModel.*
The faces are from my personal collection, except for face 04 which is from the CMU frontal test set [5].

Figure 6.8: *Landmarks generated by the TrimmedModel (continued).*

Chapter 7

Conclusions

Everyone knows the game where you ask an infant to point out your nose or your eyes. In this thesis a computer played the same game. This final chapter summarizes the methods used and suggests future avenues of research.

7.1 Summary of contributions

This project extended the classical one-dimensional Active Shape Model of Cootes et al. in various ways, cumulating in the "TrimmedModel" (section 5.10). The principal techniques, in the approximate order they were introduced, were

1. extending the set of landmarks (section 5.1)
2. using modifications such as adding noise to the training set (sections 5.4.4, 5.4.8, 5.5.8, and 5.16)
3. selectively using two instead of one-dimensional texture profiles at landmarks (section 5.5)
4. stacking two ASMs in series (section 5.7)
5. trimming covariance matrices by setting most entries to zero, for speed (section 5.10).

The result is a facial feature locater that gives better results than the classical ASM (section 6.2). On monochrome frontal upright faces with neutral expressions, it is competitive with published methods (section 6.5).

The procedures used are fairly standard. Perhaps the main contribution of this thesis was assembling them together in a sound fashion.

A few simple rules of thumb for improving ASMs emerged during this project. You get better results with a better start shape, and you can do this by running two models in series. You can

get better fits by adding more landmarks. You can discard most elements of the covariance matrices for increased speed with almost no loss of quality.

7.2 Future research

There are many ways to modify any implementation of the ASM. Chapter 3 described various techniques that have already been used by other researchers. In this section, we are more interested in some immediate extensions to this dissertation's TrimmedModel.

7.2.1 Quality of profile matches

An estimate of relative certainty in the landmark positions suggested by the profile model would be useful. Looking at Figure 6.5, we see that most jaw landmarks are "sitting in space", and the profile matches at these points must surely be weak. The final fit would probably be better if such landmarks were down-weighted when the shape model corrected the suggested shape.

A pilot implementation weighted the landmarks by the inverse of the Mahalanobis distance of the profile at the landmark. This did not give better results. A better method could probably be devised without too much head scratching, using [17, 22] as a starting point.

Another form of weighting could be used in conjunction with the above technique. This would be to scale the weight at a landmark by the uncertainty of the profile model at that landmark. Landmarks which tend to be on strong edges, such as the tip of the chin, inherently are more trustworthy than, say, a profile along the edge of the nose, where the image texture is relatively flat. Weights would be estimated from the variance of training profiles at each landmark.

7.2.2 Estimating the quality of the final fit

An estimate of the quality of the final fit would be useful. (In this dissertation we measured quality by comparing to manually landmarked faces, which would not normally be available.) The estimate could be used to trigger a retry in situations were you could take another photograph of the subject. Implementation would possibly follow naturally from the measures of quality discussed in the previous section.

7.2.3 Different kinds of profiles

There are many ways of building a profile model characterizing the nature of the image around a landmark. We evaluated several in section 5.8. In fact there are so many that choosing which ones to investigate is not easy.

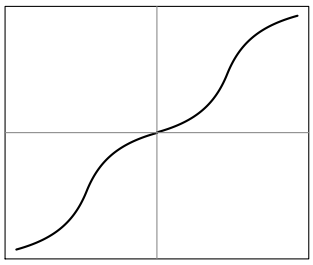


Figure 7.1: *A double sigmoid.*

The sigmoids used in section 5.8.4 reduce the effect of outliers. A better curve for inputs with both positive and negative values could be the double sigmoid shown in Figure 7.1. This would reduce the effect of small values (i.e. noise) as well as outliers.

Modern supervised classification techniques are a direction that seems to have had limited exploration by other researchers in this field. Examples are Hastie's Flexible Discriminant Analysis [39] and Friedman's Multivariate Adaptive Regression Splines [33]. A convenient listing of a number of these methods can be found in the "R Task Views" in [41, 43].

An approach that may bear fruit would be to combine or select from multiple profiling techniques at each landmark, using a decision tree or related method [7]. The thinking here is that we expect chin landmarks to be quite different from nose landmarks, and want training to automatically select the appropriate profiling techniques. So the training process would try different profiling techniques at each landmark and build a decision tree (for each landmark) that would be used to select or combine techniques during searching. This is another take on the approach of van Ginneken et al. [36]. It is a refinement of the dual profile approach tried in section 5.9, and Table 5.23 in that section gives a taste of what is possible.

7.2.4 Poor mouth location

The TrimmedModel often does a poor job of locating the subjects' mouths. Figures 6.7 and 6.8 show examples. Part of the problem is that many of the subjects are smiling in those figures, whereas the feature locater trained on the mostly neutral expressions in the XM2VTS images. But part of the problem could be that the TrimmedModel uses 1D profiles on the mouth, and 2D profiles for the other internal landmarks (section 5.5.7). Revisiting which landmarks should have 2D profiles could possibly give better mouth positions.

7.2.5 Poor worse case performance with Viola Jones start shape

Using the Viola Jones instead of the Rowley face detector gives poor worst case results (Figure 6.6). These could possibly be improved by revisiting how the start shape is generated from the Viola Jones face location.

7.2.6 Bits and pieces

We list below a few other possibilities for further work.

The model could be extended to include the whole head and not just the face. This could be done by building a larger shape model, or by combining two models: one for the face and one for the head. The outline of the head could perhaps be better modeled with a different kind of model (not an ASM).

It is usual in ASMs to reduce non-linearity by using tangent space projections [20]. Tangent spaces were not used in this project, but may produce slightly better fits. However, Stegmann [64] reports that tangent spaces had no practical effect for ASMs modeling human hands.

In section 5.10.3 we forced positive definiteness with a crude algorithm that clips negative eigenvalues. Some benefit may be obtained from alternate methods [35].

7.3 Conclusion

The Active Shape Model is one of the simplest methods for locating facial features. By focusing on the specific application of locating features in monochrome frontal upright faces with neutral expressions, this dissertation has shown that Active Shape Models can be competitive with more sophisticated methods.

On arbitrary images, computers are not yet as accurate as human landmarkers. The methods in this dissertation are one way of getting closer to that goal.

Appendix A

Face landmarks

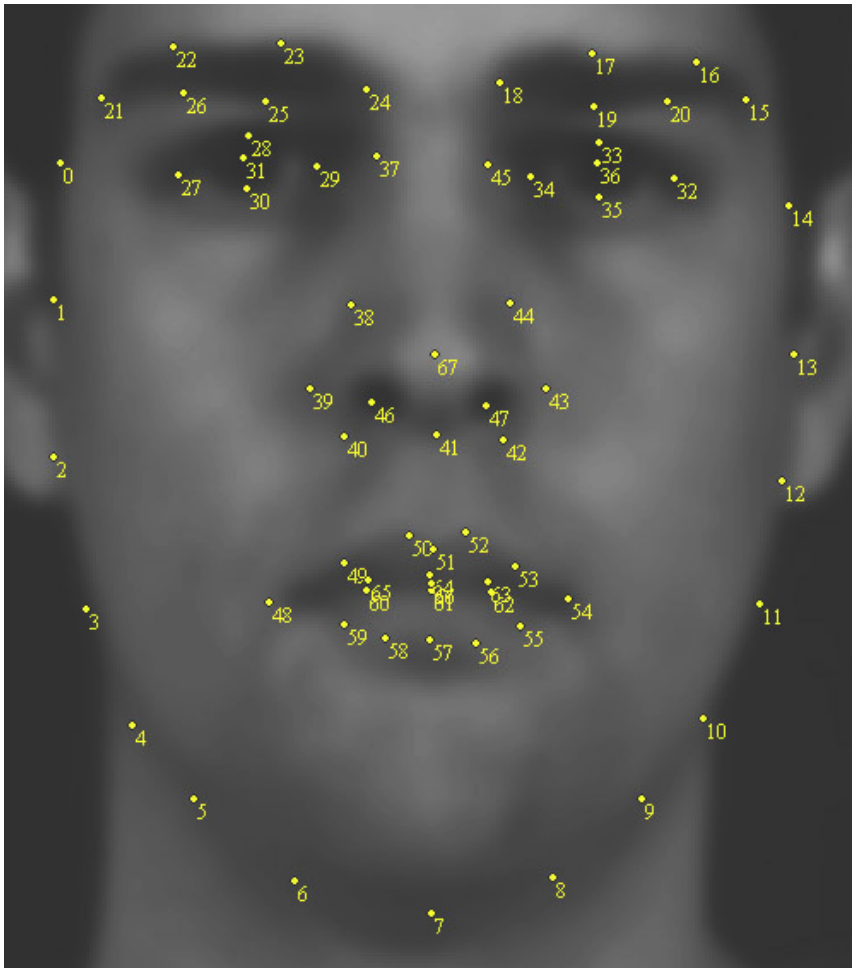


Figure A.1: *XM2VTS landmarks (0 based indexing).*

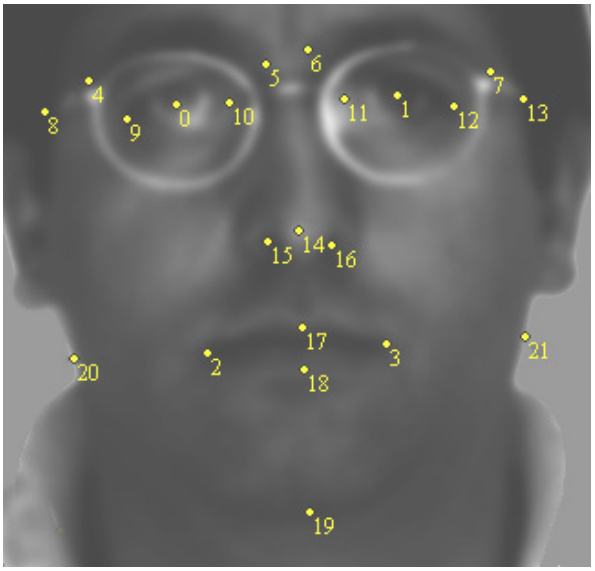


Figure A.2: *AR landmarks.*

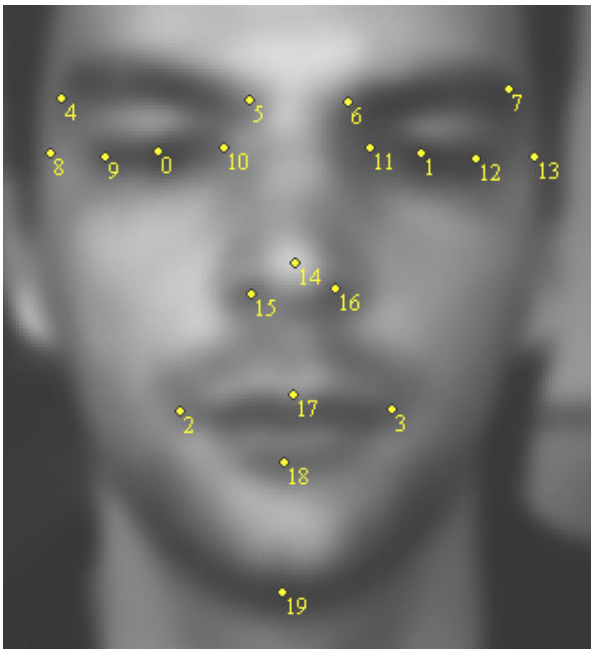


Figure A.3: *BioId landmarks.*

LANDMARK NAMES

LTemple	0	LMouthCorner	48	
LJaw1	1	Mouth49	49	
LJaw2	2	Mouth50	50	
LJaw3_MouthLine	3	MouthTopOfTopLip	51	
LJaw4	4	Mouth52	52	
LJaw5	5	Mouth53	53	
LJaw6	6	RMouthCorner	54	
TipOfChin	7	Mouth55	55	
RJaw6	8	Mouth56	56	
RJaw5	9	MouthBotOfBotLip	57	
RJaw4	10	Mouth58	58	
RJaw3_MouthLine	11	Mouth59	59	
RJaw2	12	Mouth60	60	
RJaw1	13	MouthTopOfBotLip	61	
RTemple	14	Mouth62	62	
ROuterEyeBrow	15	Mouth63	63	
ROuterTopEyeBrow	16	Mouth64	64	
RInnerTopEyeBrow	17	Mouth65	65	
RInnerEyeBrow	18	MouthBotOfTopLip	66	
Landmark19	19	NoseTip	67	last XM2VTS point
Landmark20	20	LEye0	68	left eye points created for this project
LOuterEyeBrow	21	LEye1	69	
LOuterTopEyeBrow	22	LEye2	70	
LInnerTopEyeBrow	23	LEye3	71	
LInnerEyeBrow	24	LEye4	72	
Landmark25	25	LEye5	73	
Landmark26	26	LEye6	74	
LEyeOuter	27	LEye7	75	
LEyeTop	28	REye0	76	synthesized right eye points
LEyeInner	29	REye1	77	
LEyeBottom	30	REye2	78	
LEye	31	REye3	79	
REyeOuter	32	REye4	80	
REyeTop	33	REye5	81	
REyeInner	34	REye6	82	
REyeBottom	35	REye7	83	
REye	36			
LNoseTop	37			
LNoseMid	38			
LNoseBot0	39			
LNoseBot1	40			
Nosebase	41			
RNoseBot1	42			
RNoseBot0	43			
RNoseMid	44			
RNoseTop	45			
LNostril	46			
RNostril	47			

Table A.1: *Landmark names used in this document.*
This table uses 0 based indexing. "L" and "R" stand for left and right, and are with respect to the viewer. The first 68 landmarks are the standard XM2VTS landmarks. See the source file ***atland.hpp*** *for details.*

Bibliography

[1] E. H. Adelson, C. H. Anderson, J. R. Bergen, P. J. Burt, and J. M. Ogden. *Pyramid Methods in Image Processing.* RCA Engineer 29-6, 1984. `web.mit.edu/persci/people/adelson/pub_pdfs/RCA84.pdf`.

[2] Stephan Al-Zubi. *Active Shape Structural Model (Doctoral Thesis).* Otto-von-Guericke University of Magdeburg, 2004. `citeseer.ist.psu.edu/732672.html`.

[3] Stephan Al-Zubi and Klaus Toennies. *Generalizing the Active Shape Model by Integrating Structural Knowledge to Recognize Hand Drawn Sketches.* Computer Analysis of Images and Patterns, 10th International Conference, CAIP 2003, Groningen, The Netherlands, 2003.

[4] Henri Frederic Amiel. *Le Journal Intime.* Available online, original c 1840. `en.wikipedia.org/wiki/Henri_Frederic_Amiel`.

[5] Baluja, Kanade, Poggio, Rowley, and Sung. *CMU Frontal Face Images.* Carnegie Mellon University, Robotics Institute, 1995. `vasc.ri.cmu.edu/idb/html/face/frontal_images`.

[6] J.M. Bland and D.G. Altman. *Statistical Methods for Assessing Agreement between Two Methods of Clinical Measurement.* Lancet, i, 307–310, 1986. `www-users.york.ac.uk/~mb55/meas/ba.htm`.

[7] Breiman, Friedman, Olshen, and Stone. *Classification and Regression Trees.* Wadsworth, 1984.

[8] Christopher J. C. Burges. *A Tutorial on Support Vector Machines for Pattern Recognition.* Data Mining and Knowledge Discovery 2:121–167, 1998.

[9] Hong Chen, Ying-Qing Xu, Heung-Yeung Shum, Song-Chun Zhu, and Nan-Ning Zheng. *Example-based Facial Sketch Generation with Non-parametric Sampling.* Proceedings. Eighth IEEE International Conference on Computer Vision, Volume 2, pages 433–438, 2001. `research.microsoft.com/~yqxu/papers/ICCV-2001-las-version.pdf`.

[10] William S. Cleveland. *Visualizing Data.* Hobart Press, 1993. `cm.bell-labs.com/cm/ms/departments/sia/wsc`.

[11] T. Cootes and C. Taylor. *A Mixture Model for Representing Shape Variation.* Proceedings of British Machine Vision Conference, pages 110–119, BMVA Press, 1997. `www.isbe.man.ac.uk/~bim/refs.html`.

[12] T. F. Cootes, D. H. Cooper, C. J. Taylor, and J. Graham. *A Trainable Method of Parametric Shape Description.* 2nd British Machine Vision Conference 54–61. Springer-Verlag, 1991. `www.isbe.man.ac.uk/~bim/refs.html`.

[13] T. F. Cootes, G. J. Edwards, and C. J. Taylor. *Active Appearance Models.* H.Burkhardt and B. Neumann, editors, 5th European Conference on Computer Vision, Volume 2, pages 484–498. Springer, Berlin, 1998. `www.isbe.man.ac.uk/~bim/refs.html`.

[14] T. F. Cootes, G. J. Edwards, and C. J. Taylor. *Comparing Active Shape Models with Active Appearance Models*, volume 1. Proc. British Machine Vision Conference (ed T.Pridmore, D.Elliman), 1999. `www.isbe.man.ac.uk/~bim/refs.html`.

[15] T. F. Cootes and C. J.Taylor. *Active Shape Models.* 3rd British Machine Vision Conference, pages 266–275. Springer-Verlag, 1992. `www.isbe.man.ac.uk/~bim/refs.html`.

[16] T. F. Cootes, C. J.Taylor, D. Cooper, and J. Graham. *Training Models of Shape from Sets of Examples.* 3rd British Machine Vision Conference, pages 9–18. Springer-Verlag, 1992. `www.isbe.man.ac.uk/~bim/refs.html`.

[17] T. F. Cootes and C. J. Taylor. *Active Shape Model Search using Local Grey-Level Models: A Quantitative Evaluation.* 4th British Machine Vision Conference, pages 639–648. BMVA Press, 1993. `www.isbe.man.ac.uk/~bim/refs.html`.

[18] T. F. Cootes and C. J. Taylor. *A Mixture Model for Representing Shape Variation.* Image and Vision Computing, 17(8):567–574, 1999. `www.isbe.man.ac.uk/~bim/refs.html`.

[19] T. F. Cootes and C. J. Taylor. *On Representing Edge Structure for Model Matching.* Proc. CVPR, 2001. `www.isbe.man.ac.uk/~bim/refs.html`.

[20] T. F. Cootes and C. J. Taylor. *Technical Report: Statistical Models of Appearance for Computer Vision.* The University of Manchester School of Medicine, 2004. `www.isbe.man.ac.uk/~bim/Models/app_models.pdf`.

[21] T. F. Cootes, C. J. Taylor, and A. Lanitis. *Active Shape Models: Evaluation of a Multi-resolution Method for Improving Image Search.* 5th British Machine Vision Conference, pages 327–336, York, 1994. `www.isbe.man.ac.uk/~bim/refs.html`.

[22] T. F. Cootes, C. J. Taylor, A. Lanitis, D. H. Cooper, and J. Graham. *Building and Using Flexible Models Incorporating Grey-level Information.* 4th International Conference on Computer Vision, pages 355–365. IEEE Computer Society Press, 1993. `www.isbe.man.ac.uk/~bim/refs.html`.

[23] D. Cristinacce. *Automatic Detection of Facial Features in Grey Scale Images (Doctoral Thesis).* University of Manchester (Faculty of Medicine, Dentistry, Nursing and Pharmacy), 2004. `mimban.smb.man.ac.uk/publications/index.php`.

[24] D. Cristinacce and T. Cootes. *A Comparison of two Real-Time Face Detection Methods.* 4th IEEE International Workshop on Performance Evaluation of Tracking and Surveillance, 2003. `mimban.smb.man.ac.uk/publications/index.php`.

[25] D. Cristinacce and T. Cootes. *Facial Feature Detection using AdaBoost with Shape Constraints*. 14th British Machine Vision Conference, Norwich, 2003. `mimban.smb.man.ac.uk/publications/index.php`.

[26] D. Cristinacce and T. Cootes. *A Comparison of Shape Constrained Facial Feature Detectors*. 6th International Conference on Automatic Face and Gesture Recognition 2004, Seoul, 2004. `mimban.smb.man.ac.uk/publications/index.php`.

[27] D. Cristinacce and T. Cootes. *Facial Feature Detection and Tracking with Automatic Template Selection*. 7th International Conference on Automatic Face and Gesture Recognition 2006, Southampton, 2006. `mimban.smb.man.ac.uk/publications/index.php`.

[28] D. Cristinacce and T. Cootes. *Feature Detection and Tracking with Constrained Local Models*. 17th British Machine Vision Conference, Edinburgh, 2006. `mimban.smb.man.ac.uk/publications/index.php`.

[29] D. Cristinacce, T. Cootes, and I. Scott. *A Multi-Stage Approach to Facial Feature Detection*. 15th British Machine Vision Conference, London, 2004. `mimban.smb.man.ac.uk/publications/index.php`.

[30] I. L. Dryden and Kanti V. Mardia. *Statistical Shape Analysis*. Wiley, 1998. `www.maths.nott.ac.uk/personal/ild/book/index.html`.

[31] R. O. Duda, P. E. Hart, and D. G. Stork. *Pattern Classification*. Wiley-Interscience Publication, 2000. `www.crc.ricoh.com/~stork/DHS.html`.

[32] G. Edwards, C. J. Taylor, and T. F. Cootes. *Interpreting Face Images Using Active Appearance Models*. 3rd International Conference on Automatic Face and Gesture Recognition 1998, pages 300–305, Japan, 1998.

[33] J. H. Friedman. *Multivariate Adaptive Regression Splines (with discussion)*. Annals of Statistics 19, 1, 1991. `www-stat.stanford.edu/~jhf`.

[34] M. Galassi and et al. *GNU Scientific Library Reference Manual*. GNU, 2006. `www.gnu.org/software/gsl`.

[35] James E. Gentle. *Numerical Linear Algebra for Applications in Statistics*. Springer, 1998. See page 178 for methods of forcing positive definiteness. `mason.gmu.edu/~jgentle`.

[36] B. van Ginneken, A.F.Frangi, J.J.Stall, and B. ter Haar Romeny. *Active Shape Model Segmentation with Optimal Features*. IEEE-TMI, 21:924–933, 2002.

[37] R. C. Gonzalez and R. E. Woods. *Digital Image Processing, 2nd Edition*. Prentice Hall, 2002. `www.imageprocessingplace.com`.

[38] C. Harris and M. Stephens. *A Combined Corner and Edge Detector*. Alvey Vision Conference, pages 147–151, 1988. `www.csse.uwa.edu.au/~pk/research/matlabfns/Spatial/Docs/Harris` or `www.engr.udayton.edu/faculty/jloomis/ece595b/notes/Harris_Corner_Algorithm/corners.html`.

[39] Hastie, Tibshirani, and Buja. *Flexible Discriminant Analysis by Optimal Scoring*. Journal of The American Statistical Association, 1255–1270, 1994. `www-stat.stanford.edu/~hastie/pub.htm`.

[40] T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer, 2003. `www-stat.stanford.edu/~tibs/ElemStatLearn`.

[41] Paul Hewson. *(maintainer) CRAN Task View: Multivariate Statistics*. R language online document, 2007. `http://cran.r-project.org/src/contrib/Views/Multivariate.html`.

[42] Erik Hjelmås and Boon Keen Low. *Face Detection: A Survey*. Computer Vision and Image Understanding, 83(3), pages 236–274, 2001.

[43] Torsten Hothorn. *(maintainer) CRAN Task View: Machine Learning and Statistical Learning*. R language online document, 2007. `http://cran.r-project.org/src/contrib/Views/MachineLearning.html`.

[44] O. Jesorsky, K. Kirchberg, and R. Frischholz. *Robust Face Detection using the Hausdorff Distance*. J. Bigun and F. Smeraldi, editors, Audio and Video based Person Authentication - AVBPA. Springer, 2001. `www.bioid.com/downloads/facedb`.

[45] Richard A. Johnson and Dean W. Wichern. *Applied Multivariate Statistical Analysis, 5th Edition*. Prentice Hall, 2002. `http://vig.prenhall.com/catalog/academic/product/0,1144,0130925535,00.html`.

[46] Michael Kass, Andrew Witkin, and Demetri Terzopoulos. *Snakes: Active contour models*. International Journal of Computer Vision, 1:321–331, 1987.

[47] J. J. Koenderink and A. J. van Doorn. *The Structure of Locally Orderless Images*. Int. J. Comput. Vis., Volume 31, no. 2/3, pages 159–168, 1999.

[48] Yuanzhong Li and Wataru Ito. *Shape Parameter Optimization for AdaBoosted Active Shape Model*. Tenth IEEE International Conference on Computer Vision (ICCV'05), Volume 1, pages 251–258, 2005.

[49] Rainer Lienhart and Jochen Maydt. *An Extended Set of Haar-like Features for Rapid Object Detection*. Submitted to ICIP2002, 2002.

[50] A.M. Martinez and R. Benavente. *The AR Face Database*. CVC Tech. Report 24, 1998. `rvl1.ecn.purdue.edu/~aleix/aleix_face_DB.html`.

[51] K. Messer, J. Matas, J. Kittler, J. Luettin, and G. Maitre. *XM2VTS: The Extended M2VTS Database*. Proceedings 2nd Conference on Audio and Video-base Biometric Personal Verification (AVBPA99) Springer Verlag, New York, 1999. `www.ee.surrey.ac.uk/Research/VSSP/xm2vtsdb`.

[52] M.Rogers and J.Graham. *Robust Active Shape Model Search*. 7th European Conference on Computer Vision, Volume 4, pages 517–530. Springer, 2002.

[53] 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.

[54] E. Persoon and K. S. Fu. *Shape Discrimination using Fourier Descriptors*. IEEE Trans. Syst., Man, Cybern. Volume SMC–7, pages 170–179, 1977.

[55] FGNET project. *The AR Face Database 22 Points Markup*. FGNET, 1998. `www-prima.inrialpes.fr/FGnet/data/05-ARFace/tarfd_markup.html`.

[56] Intel Research. *Open Source Computer Vision Library*. Intel, 2007. `www.intel.com/research/mrl/research/opencv`.

[57] S. Romdhani, S. Gong, and A. Psarrou. *A Multi-view Non-linear Active Shape Model using Kernel PCA*. 10th British Machine Vision Conference Volume 2, pages 483–492, Nottingham, BMVA Press, 1999.

[58] Henry A. Rowley, Shumeet Baluja, and Takeo Kanade. *Neural Network-Based Face Detection*. IEEE Transactions on Pattern Analysis and Machine Intelligence, Volume 20, pages 23–38, 1998. `vasc.ri.cmu.edu/NNFaceDetector`.

[59] S. Scholkopf, A. Smola, and K. Muller. *Nonlinear Component Analysis as a Kernel Eigenvalue Problem*. Neural Computation 10(5) 1299–1319, 1998.

[60] I. M. Scott, T. F. Cootes, and C. J. Taylor. *Improving Appearance Model Matching Using Local Image Structure*. Information Processing in Medical Imaging, 18th International Conference, 2003. `www2.wiau.man.ac.uk/caws/Conferences/10/proceedings/8/papers/41/ipmi2003.pdf`.

[61] Christopher G. Small. *The Statistical Theory of Shape*. Springer, 96.

[62] Lindsay I Smith. *A Tutorial on Principal Components Analysis*. Available online, 2002. `csnet.otago.ac.nz/cosc453/student_tutorials/principal_components.pdf`.

[63] 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.

[64] M. B. Stegmann. *A Brief Introduction to Statistical Shape Analysis*. Available online, 2002. `www2.imm.dtu.dk/pubdb/views/edoc_download.php/403/pdf/imm403.pdf`.

[65] M. B. Stegmann. *Generative Interpretation of Medical Images (Doctoral Thesis)*. Informatics and Mathematical Modelling, Technical University of Denmark, DTU, 2004. `www2.imm.dtu.dk/pubdb/p.php?3126`.

[66] R Development Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, 2007. `www.R-project.org`.

[67] P. Viola and M. Jones. *Rapid object detection using a boosted cascade of simple features*, volume 1. Computer Vision and Pattern Recognition Conference, 2001. `citeseer.ist.psu.edu/article/viola01rapid.html`.

[68] A. L. Yuille, P. W. Hallinan, and D. S. Cohen. *Feature extraction from faces using deformable templates*. Int. J. Comput. Vision 8, 1992, 99–111, 1992. `www.stat.ucla.edu/~yuille`.

[69] C. Zahn and R. Roskies. *Fourier Descriptors for Plane Closed Curves*. IEEE Transactions on Computers C–21 269–281, 1972.

[70] Y. Zhou, L. Gu, and H.J. Zhang. *Bayesian Tangent Shape Model: Estimating Shape and Pose Parameters via Bayesian Inference*. Computer Vision and Pattern Recognition, 2003. `facealignment.ius.cs.cmu.edu/alignment/webdemo.html`.