

Automatic Detection of Facial Features in Grey Scale Images

A thesis submitted to the University of Manchester
for the degree of Doctor of Philosophy
in the Faculty of Medicine, Dentistry, Nursing and Pharmacy

2004

David Cristinacce

Imaging Science and Biomedical Engineering

Contents

1	Introduction	24
1.1	Facial Feature Detection	24
1.2	Motivations	25
1.3	Challenges	25
1.4	Approach	26
1.5	Structure of Thesis	28
2	Face and Facial Feature Detection Background	30
2.1	Scope of Literature Review	30
2.2	Template Based Methods	31
2.3	Multi-view Template Methods	39
2.4	Feature Based Methods	46
2.5	Feature Localisation Methods	50
2.6	Iterative Search Methods	54
2.7	Summary	57
3	Face Detection Methods	59
3.1	Face Detection Training Set	59

3.2	Linear Profile Detector (LPD)	60
3.2.1	Linear Profile Models	60
3.2.2	LPD Training	61
3.2.3	LPD Search	62
3.3	Orientation Map Detector (OMD)	64
3.3.1	Orientation Maps	64
3.3.2	Comparing two Orientation Maps	65
3.3.3	OMD Training	66
3.3.4	OMD Search	67
3.4	Normalised Correlation Detector (NCD)	68
3.4.1	Comparing Images using Normalised Correlation	68
3.4.2	NCD Training	68
3.4.3	NCD Search	69
3.5	Boosted Cascade Detector (BCD)	70
3.5.1	Integral Images	70
3.5.2	Feature Classifiers	71
3.5.3	BCD - Training with AdaBoost	72
3.5.4	BCD - Search with AdaBoost Model	74
3.5.5	BCD - Training the Cascade	76
3.5.6	BCD - Cascaded Search	78
3.6	Face Detection Test Sets	79
3.6.1	XM2VTS	79
3.6.2	BIOID	80

3.6.3	CMU	81
3.7	Face Detection Testing Methodology	81
3.7.1	Face Localisation Accuracy	82
3.7.2	FROC Curve Analysis	83
3.8	Face Detection Results	85
3.8.1	XM2VTS	85
3.8.2	BIOID	86
3.8.3	CMU	87
3.9	Timings	89
3.10	Conclusions	90
4	Boosted Cascade Detector Experiments	91
4.1	Benchmark Detector	91
4.2	Comparison with Single AdaBoost Model	94
4.3	Vary Number of Features	95
4.4	Vary Training Set Size	96
4.5	Vary Template Resolution	97
4.6	Vary Face Region	99
4.7	Single Feature Detectors	100
4.8	Extensions to the BCD Literature Review	102
4.9	Conclusions	105
5	Shape Modelling	107
5.1	Suitable Landmarks	107
5.2	Intrinsic Shape Variation	108

5.3	Aligning the Training Data	109
5.4	Building a Shape Model	110
5.5	Fitting to New Points	113
5.5.1	Fitting to a Subset of Points	113
5.6	Likelihood of a Shape	115
5.7	Conclusions	118
6	Shape Constrained Feature Detection	119
6.1	Local Feature Detectors	119
6.1.1	Training Data	120
6.1.2	Individual Feature Detectors	120
6.1.3	Unconstrained Feature Search	121
6.1.4	Proximity Measure	122
6.1.5	Unconstrained Search Results	123
6.1.6	Unconstrained Search Examples	125
6.1.7	Unconstrained Search Timing	126
6.1.8	Unconstrained Search Conclusions	126
6.2	Combinatoric Shape Search (CSS)	127
6.2.1	Probability of Individual Feature Responses	127
6.2.2	Shape Constraints	128
6.2.3	Point Set Objective Function	129
6.2.4	Finding the Best Candidate Point Set	129
6.2.5	CSS Results	131
6.2.6	CSS Example Searches	133

6.2.7	CSS Timing	134
6.2.8	CSS Conclusions	134
6.3	Shape Optimised Search (SOS)	135
6.3.1	Feature Response Images	135
6.3.2	Non Linear Optimisation	136
6.3.3	SOS Results	137
6.3.4	SOS Search Examples	139
6.3.5	SOS Timing	140
6.3.6	SOS Conclusions	141
6.4	Pairwise Reinforcement of Feature Responses (PRFR)	141
6.4.1	Pairwise Feature Distributions	141
6.4.2	PRFR Search	144
6.4.3	PRFR Results	146
6.4.4	PRFR Search Examples	148
6.4.5	PRFR Timing	149
6.4.6	PRFR Conclusions	149
6.5	Comparison of SOS and PRFR Algorithms	150
6.5.1	SOS and PRFR 17pt Results	150
6.5.2	SOS and PRFR Search Examples	152
6.5.3	Individual Feature Errors	153
6.6	Conclusions	154
7	Active Appearance Models	156
7.1	Appearance Models (APMs)	156

7.1.1	Shape Model	156
7.1.2	Texture Model	157
7.1.3	Combined Model	158
7.2	Active Appearance Models (AAMs)	160
7.2.1	AAM Search	160
7.2.2	AAM Training	161
7.3	AAM Results	162
7.3.1	Vary Model Resolution	162
7.3.2	Vary Region Modelled	164
7.3.3	Vary Sampling Method	165
7.3.4	Optimal AAM Formulation	168
7.4	AAM Timings	169
7.5	AAM Conclusions	170
7.6	Combining Feature Detection and the AAM	170
7.6.1	Comparing AAM, SOS and PRFR	170
7.6.2	Combined PRFR and AAM Search	172
7.6.3	Individual Feature Accuracy	173
7.6.4	PRFR+AAM Timings	174
7.6.5	PRFR+AAM Search Conclusions	175
7.7	Comparison with Other Published Results	175
7.8	Comparison with Human Landmarking	177
7.9	Conclusions	180
8	Discussion and Conclusions	182

8.1	Summary of Thesis	182
8.2	Discussion	184
8.3	Future work	186
8.4	Final Statement	188
A	Aligning Two 2D Shapes	189
B	Weighted fitting of PCA parameters	191
C	BCD Training Set	193
D	Facial Feature Detector Training Examples	194
E	Normalised Correlation Feature Templates	196
F	Orientation Map Feature Templates	197
G	Features selected by AdaBoost	198
H	PRFR Histograms	200
I	Human Landmark Test Images	202
J	Average Human Landmarks vs PRFR+AAM	204

List of Figures

1.1	Example labelled image with 17 landmark points	24
1.2	Examples of individual feature detections (white crosses) within the region predicted by a face detector (white box). Black crosses are manually labelled ground truth feature locations.	27
2.1	System diagram for the Rowley Detector, reproduced with permission from [74]	35
2.2	Example of image compression using various numbers of Gabor Wavelets, reproduced with permission from [23]	50
2.3	Overview of face localisation using the Hausdorff distance, reproduced with permission from [43]	52
2.4	Examples of valid and invalid feature detections	56
3.1	Example WEBCAM training images with 20 landmark points	60
3.2	Orientation map of the average face	66
3.3	The integral image	70
3.4	Different feature types	71
3.5	Features selected by AdaBoost, overlaid on an example from the training set	74
3.6	Boosted Cascade Detector multi-scale search parameters, start scale s , step size t and scale increment i_s	75

3.7	Cascaded search, each level returns either true or false. If a subwindow fails a level then no more levels are evaluated. The final score is $c_s = \sum_{i=1} s_i$, where s_i is the AdaBoost response* from level i	79
3.8	Example XM2VTS images	80
3.9	Example BIOID images	80
3.10	Examples of images <i>not</i> included in the BIOID subset, as they do not include the whole face region.	80
3.11	Example CMU images	82
3.12	Distance metric between the eye points predicted by the candidate region and the true eye points $d_{eyes} = \frac{R+L}{2S}$	83
3.13	The distance metric between a candidate defined by corners (A,B) and a candidate defined by corners (C,D) $d_{cands} = \frac{d_{AC}^2 + d_{BD}^2}{d_{AB}^2 + d_{CD}^2}$, where d_{AB} is the distance between points A & B.	85
3.14	Proportion of faces found at different thresholds for d_{eyes} , when searching the XM2VTS test set	86
3.15	Proportion of faces found at different thresholds for d_{eyes} , when searching the BIOID test set	87
3.16	FROC curve for the Boosted Cascade Detector and Orientation Map Detector when applied to the CMU test set	88
3.17	Example search results on the CMU images	89
4.1	Reflecting and over-sampling a face image	92
4.2	Features selected by AdaBoost, overlaid on an example from the training set	93
4.3	Proportion of successful searches at different thresholds for d_{eyes} , when searching with the (16-level 24x24 pixel) Boosted Cascade Detector from Chapter 3 and the new (10-level 21x21 pixel) benchmark detector	94
4.4	Proportion of successful searches at different thresholds for d_{eyes} , when searching with the single template and cascade	95

4.5	Effect of varying the number of features in each level	96
4.6	Varying the training set size and resampling	97
4.7	Varying face template resolution	98
4.8	Varying the template resolution	99
4.9	Cropped face regions at various resolutions	99
4.10	Cropping the face template region	100
4.11	Varying right eye template region	101
4.12	Comparing whole face detection with various right eye detectors . . .	101
4.13	Expanded feature set used by Lienhart <i>et al.</i> [53]	103
4.14	Feature types used by Li <i>et al.</i> [49]	104
5.1	Corresponding landmark points between 2 examples from the WEB-CAM data set	108
5.2	Aligning a set of shapes	110
5.3	First two modes of the human face shape model	112
5.4	Example log likelihoods when fitting to 20 face points (denoted by x) after applying various distortions	116
5.5	Example log likelihoods when fitting to 4 face points, (both pupils and both mouth corners) after applying various distortions	117
6.1	Example feature training images	120
6.2	Proximity Measure, “.” = true feature location and “x”= predicted feature location	122
6.3	Point to point error when performing unconstrained feature detection on the BIOD test set	124
6.4	Examples of unconstrained search with Boosted Cascade Detectors .	125

6.5	Average mean positional error (m_{e4}) when performing CSS on the BIOID test set	132
6.6	Examples of improvement in search accuracy (m_{e4}) when using CSS compared to unconstrained search, using Boosted Cascade Detectors .	133
6.7	Response images for the eyes and mouth corner detectors (black implies strong response)	136
6.8	Average mean positional error (m_{e4}) when performing SOS on the BIOID test set [†]	138
6.9	Examples of improvement in search accuracy (m_{e4}) using SOS 17pt search compared to CSS 4pt search with Boosted Cascade Detectors .	139
6.10	Right eye pupil location histograms relative to the best match of four different feature detectors (black pixels indicate peaks in each histogram)	142
6.11	Probability density distributions d_{ijt} in the histogram frame for feature point i predicted by relative histogram h_{ij} given the two best matches of detector j	145
6.12	Average mean positional error (m_{e4}) when performing PRFR on the BIOID test set	147
6.13	Examples of improvement in search accuracy m_{e4} using PRFR 17pt search, compared to CSS 4pt search with Boosted Cascade Detectors.	148
6.14	Comparison of PRFR and SOS algorithms on the BIOID test set given the same feature detector, using the average mean positional error of all 17pts (m_{e17})	151
6.15	Comparing performance of PRFR and SOS algorithms with different feature detectors on the BIOID test set, using the average mean positional error of all 17pts (m_{e17})	152
6.16	Examples of SOS and PRFR 17pt search with different feature detectors	153
6.17	Average point to point errors m_{e1} for each feature when performing PRFR, SOS and average point prediction on the BIOID test set (error bars show upper and lower quartiles of m_{e1} for each feature).	154

7.1	Modes of Shape Model ($\pm 2.5\text{std}$)	157
7.2	Modes of Texture Model ($\pm 2.5\text{std}$)	158
7.3	Modes of Combined Model ($\pm 2.5\text{std}$)	160
7.4	Mean face of APMs, built at different resolutions	163
7.5	Point to point error m_{e17} when performing AAM search on the BIOID test set, varying the resolution of the APM	163
7.6	Two different labelling schemes	164
7.7	Modes of Combined Model 22pts, sampling 1000 pixels ($\pm 2.5\text{std}$)	165
7.8	Point to point error m_{e17} when performing AAM search on the BIOID test set, varying the face region modelled by the APM	166
7.9	Point to point error m_{e17} when performing AAM search on the BIOID test set, varying the texture sampling method	167
7.10	Point to point error m_{e17} when performing AAM search on the BIOID test set using corner texture sampling	168
7.11	Comparing PRFR, SOS and AAM point to point error m_{e17} when searching the BIOID test set	171
7.12	PRFR+AAM Search Error m_{e17} when searching the BIOID test set	172
7.13	Point to point error m_{e1} for each feature when performing PRFR and PRFR+AAM searches on the BIOID test set	174
7.14	Eye pupil finding comparison on the XM2VTS [59] and BIOID test sets [43]	176
7.15	Bar chart to show average point to point error (m_{e17}) for each of the 10 test images, using human landmarks and points predicted using PRFR+AAM search.	178
7.16	Bar chart to show average point to point error for various features, using human landmarks and points predicted using PRFR+AAM. The error bars show the range of positional error m_{e1} for each feature.	180

List of Tables

2.1	Comparison of detection rate versus number of false detections for various algorithms applied to the MIT data set (subset B of the CMU data set).	34
2.2	Comparison of detection rate versus number of false detections for various algorithms applied to the CMU data set, (expanded version of table appearing in [93]).	36
2.3	Comparison of detection rate versus number of false detections for various algorithms applied to the CMU in-plane rotations data set. .	40
2.4	Comparison of detection rate versus number of false detections for various algorithms applied to the CMU out-of-plane profile data set, (expanded version of table appearing in [97]).	42
3.1	Classification of true and false positives. Based on the d_{eyes} metric .	84
3.2	Time to search BIOID image (384*286 pixels) using a 500Mhz PII processor	90
4.1	Number of possible features during training, given different size templates	98
6.1	Time to perform unconstrained feature search on a single BIOID image using a 500Mhz PII processor	126
6.2	Time to perform CSS with BCD detectors on a single BIOID image using a 500Mhz PII processor	134
6.3	Time to perform SOS with BCDs on a single BIOID image using a 500Mhz PII processor	140

6.4	Time to perform PRFR with BCDs on a single BIOID image using a 500Mhz PII processor	149
7.1	Time to perform various AAM searches on a single BIOID image using a 500Mhz PII processor	169
7.2	Time to perform PRFR+AAM search on a single BIOID image, using a 500Mhz PII processor	174

List of Algorithms

2.1	Boot-strap Training [87]	33
3.1	Linear Profile Model Training Method	62
3.2	Testing Candidate Pairs	64
3.3	Orientation Map Detector Coarse-to-Fine Search	67
3.4	Variance Normalisation of an Image	69
3.5	AdaBoost - Adaptive Boosting (based on [92])	73
3.6	Efficient Variance Normalisation	76
3.7	Boosted Cascade Detector- Building the Cascade	77
5.1	Aligning the Training Data [10]	109
5.2	Principal Components Analysis	111
5.3	Iterative Fitting to New Points [10]	114
6.1	Iterative Search Method	130

List of Abbreviations

- AAM = Active Appearance Model
- APM = Appearance Model
- ASM = Active Shape Model
- BCD = Boosted Cascade Detector
- CSS = Combinatoric Shape Search
- FROC = Free Receiver Operator Characteristic
- GMM = Gaussian Mixture Model
- GWN = Gabor Wavelet Network
- MLP = Multi-Layer Perceptron
- OMD = Orientation Map Detector
- NCD = Normalised Correlation Detector
- PCA = Principal Components Analysis
- PRFR = Pairwise Reinforcement of Feature Responses
- SNoW = Sparse Network of Winnows
- SOS = Shape Optimised Search
- SVM = Support Vector Machine

Abstract

Accurate localisation of faces and facial features within grey scale images is a challenging task due to the high variability, in both shape and texture, of the appearance of the human face. This thesis investigates methods of combining shape modelling techniques and texture based pattern recognition to reliably and accurately detect facial features, such as the eye pupils, nostrils and mouth corners.

Individual feature detectors designed to find specific facial features, e.g. the right mouth corner, are found to be unreliable. The lack of distinctive local image structure around many facial features results in many false matches. Local variation in appearance due to expression, blinking or occlusion may mean the true feature is not detected at all.

These problems are addressed in two ways. Firstly, a coarse-to-fine approach is adopted to find the whole face and restrict the search region for individual features. Secondly, shape information is used to select the most likely looking group of candidate features that form a plausible face shape. Three methods of combining shape and feature detection are presented. All three methods are found to give superior performance, compared to merely selecting the best match by each feature detector.

The best performing shape constrained feature detection method is compared with the well known Active Appearance Model (AAM) approach. Shape constrained feature detection is found to outperform the basic AAM algorithm. However, a recent variation of the AAM which is tuned to edge and corner features is found to give

similar results to shape constrained feature detection.

The most accurate feature detection performance is achieved using a hybrid approach. This uses shape constrained feature detection to predict initial feature points. These feature points are then refined using edge/corner AAM search. This method is found to be comparable with the accuracy of human annotation.

Declaration

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

Copyright

1. Copyright in text of this thesis rests with the Author. Copies (by any process) either in full, or of extracts, may be made **only** in accordance with instructions given by the Author and lodged in the John Rylands University Library of Manchester. Details may be obtained from the Librarian. This page must form part of any such copies made. Further copies (by any process) of copies made in accordance with such instructions may not be made without permission (in writing) of the Author.
2. The ownership of any intellectual property rights which may be described in this thesis is vested in the University of Manchester, subject to any prior agreement to the contrary, and may not be made available for use by third parties without the written permission of the University, which will prescribe the terms and conditions of any such agreement.
3. Further information on the conditions under which disclosures and exploitation may take place is available from the Head of the Division of Imaging Science and Biomedical Engineering.

About the Author

David Cristinacce received an MA in Mathematics from the University of Cambridge in 1997 (class 2:1). He then spent two years working for an actuarial consultancy, before going back to education and received an MSc in Cognitive Science from the University of Manchester in 2000. In October 2000, he commenced research on a PhD, in the department of Imaging Science and Biomedical Engineering, at the University of Manchester, for which this thesis is submitted. During this period he published the following papers related to the work in this thesis.

- D. Cristinacce and T. Cootes. A Comparison of two Real-Time Face Detection Methods. In *Proceedings of 4th IEEE International Workshop on Performance Evaluation of Tracking and Surveillance*, pages 1–8, Graz, Austria, April 2003.
- D. Cristinacce and T. Cootes. Facial Feature Detection using AdaBoost with Shape Constraints. In *Proceedings of 14th BMVA British Machine Vision Conference*, pages 231–240, Norwich, UK, September, 2003.
- D. Cristinacce and T. Cootes. A Comparison of Shape Constrained Facial Feature Detectors. In *Proceedings of 6th IEEE International Conference on Automatic Face and Gesture Recognition*, pages 375–380, Seoul, Korea, May, 2004.
- D. Cristinacce, T. Cootes and I. Scott. A Multi-Stage Approach to Facial Feature Detection. In *Proceedings of 15th BMVA British Machine Vision Conference*, pages 277–286, London, UK, September, 2004.

Acknowledgements

I would like to thank the following people:-

My supervisor Tim Cootes for his guidance, support and enthusiasm during my research, the use of his APM+AAM source code and helpful comments whilst writing this thesis.

Kola Babalola for his help marking up the BIOID face images, which provide ground truth for many experiments throughout this thesis.

Ian Scott for use of his edge/corner AAM texture sampling code and outstanding contribution to my C++ education.

Vladimir Petrovic, Franck Bettinger, John Kang, Gavin Wheeler and Panachit Kittipanya-ngam for many helpful discussions within the faces group.

My room mates Patrick Cherry, Rhodri Davies, Hamied Haroon, John Carr, Roy Schestowitz who have given practical help and/or kept me entertained over the last three and a half years.

Paola Triolo for providing welcome distraction during the write up of this thesis.

Chapter 1

Introduction

1.1 Facial Feature Detection

The task addressed by this thesis is that of automatically locating features on the human face. For example given an unlabelled image as shown in Figure 1.1(a), the computer is asked to mark the locations of seventeen features, as shown in Figure 1.1(b).



(a) Unlabelled Image

(b) Manually Labelled Image

Figure 1.1: Example labelled image with 17 landmark points

The image in Figure 1.1(b) was landmarked manually by a human operator. However,

manual landmarking is time consuming, tedious and error prone. The algorithms in this thesis aim to automate this process.

1.2 Motivations

Some applications of automatic facial feature detection are as follows:-

- Animation - Computer generated facial expressions can be animated by tracking features on the human face and then replacing the facial texture with a cartoon like face. Therefore automatic landmarks would be useful in the computer game and film industries.
- Expression Recognition - The location of key feature points on the face could be used to aid the recognition of human facial expressions. e.g. The corners of the mouth when attempting to recognise a smile.
- Face Recognition - Algorithms that identify human beings from photographs require accurate registration to compare two given faces.
- Face Processing - Any process applied to an image containing a face requires the accurate localisation of the face before further processing takes place.

1.3 Challenges

The challenges which make facial feature detection a difficult task are common to many computer vision problems, especially general object recognition. Some face specific problems are as follows:-

- Identity variation - Human faces vary greatly between individuals

- Expression variation - One human face is capable of a great deal of variety, e.g. when blinking or opening the mouth
- Head rotation - Both in plane and out of plane rotation of the head causes major changes in visual appearance
- Lighting variation - The lighting of a face causes non-linear effects on the value of image pixels
- Scale variation - The face can appear at a wide range of sizes
- Occlusion - Facial hair or glasses can cause the facial features to be obscured.
- False positives - Background regions of the image may resemble human faces and may lead to false detections
- Speed Constraints - Face detection is usually followed by further processing, e.g. face recognition, so should ideally be an efficient real-time algorithm.

Any one of the above problems may cause a face detector or facial feature finding algorithm to fail. In general, the difficulty of face and feature detection depends largely on the data set used and the extent to which these variations are controlled.

In this thesis, the feature finding methods are restricted to static grey scale images. This makes the task more difficult because skin colour information cannot be used to aid face detection [57] and there are no motion cues as when using video input instead of single images.

1.4 Approach

In this thesis, the most effective approach to facial feature detection is found to be a coarse-to-fine approach. This effectively splits the problem into two stages:-

1. Locate the face using a face detector
2. Find facial features within the region indicated by the face detector

To solve stage 1, there are many face detection methods devised by previous authors (see review in Chapter 2). Four methods are implemented and tested in this thesis (see Chapter 3). The region supplied by best performing face detector is then taken as the starting point for testing algorithms in stage 2.

To solve stage 2 and find features given a face region, a simple approach is to reuse the face detector method, but train the detector on individual features, instead of the whole face. For example, train a detector on a small image patch surrounding the corner of the mouth. However, this method is found to give very poor results.

To improve the performance of local feature detection several algorithms are described which apply shape constraints to the output of individual feature detectors. It is found that shape constraints are fundamental to any robust method of feature point prediction. For example Figure 1.2, shows the improvement when using shape constraints compared to unconstrained feature detection.

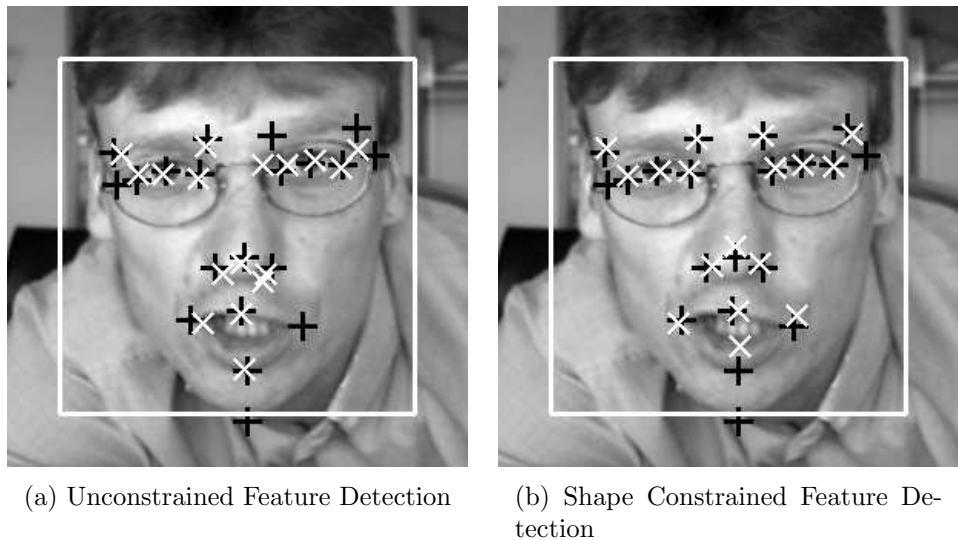


Figure 1.2: Examples of individual feature detections (white crosses) within the region predicted by a face detector (white box). Black crosses are manually labelled ground truth feature locations.

Figure 1.2(a) shows that unconstrained feature detection finds the correct feature in some cases, but fails badly in others. For example in Figure 1.2(a), the right eye region is located accurately. However on the same face, the left eye brow detectors fail and match to the glasses. The left mouth corner detector fails even more spectacularly, matching to the left nostril.

If feature detections are constrained to form a plausible face shape then these false matches can be avoided. Figure 1.2(b) shows the feature points output by a shape constrained search algorithm described in this thesis (see Section 6.3) using the same set of feature detectors.

Another approach, that can be used to perform local search given an approximate face region, is the well known Active Appearance Model (AAM) search algorithm, which was developed at Manchester University by Edwards *et al.* [22]. The AAM is compared with the shape constrained feature search methods in Chapter 7 of this thesis. Hybrid searches which use shape constrained detection to initialise the AAM are also evaluated and found to give superior results to any other method used in isolation.

1.5 Structure of Thesis

The structure of the thesis is as follows.

Chapter 2 : Face and Feature Detection Background Previous work on face detection and feature localisation is discussed.

Chapter 3 : Face Detection Methods Four approximately real-time face detection methods are described, implemented by the author and tested on three separate data sets.

Chapter 4 : Boosted Cascade Detector Experiments The Boosted Cascade

Detector is investigated in more detail.

Chapter 5 : Shape Modelling Shape modelling techniques are discussed, as shape is a useful constraint, when using noisy local feature detectors.

Chapter 6 : Shape Constrained Feature Detection Three types of feature detector and three different types of shape constraint are described and tested.

Chapter 7 : Active Appearance Models The well known Active Appearance Model (AAM) algorithm is described and combined with the shape constrained methods described in Chapter 6.

Chapter 8 : Conclusions Discussion of the research and suggestions for further work.

Chapter 2

Face and Facial Feature Detection Background

This chapter reviews previous work on face and facial feature detection. Template methods, which classify all subwindows of the image as face/non-face regions are discussed in Section 2.2 and extensions to multi-view face detection described in Section 2.3. Feature based approaches to face detection are considered in Section 2.4. Feature localisation, which aims to find features given the approximate location of the whole face, is discussed in Section 2.5. Finally iterative search algorithms are considered in Section 2.6.

2.1 Scope of Literature Review

This review of face and feature detection methods is restricted to static grey scale images, as these are the conditions under which our system is required to operate. Therefore methods that require a known static background, use motion cues or colour information are not considered. Static grey-scale methods are compared using the following criteria:-

1. Face detection performance (detection rate versus number of false positives)
2. Accuracy and reliability of feature localisation
3. Speed of detection
4. Ability to cope with rotated faces (in plane+out of plane)

The lack of a common test set often makes different methods difficult to compare. Even when a common test set is used detection systems can be biased differently between finding all true faces and avoiding false detections. Face detection methods are sometimes designed to perform slightly different tasks. e.g. Some methods detect rotated faces, while others are restricted to upright faces. Feature localisation algorithms are even harder to compare, as researchers rarely use the same data sets and do not always search for the same features. However, some data sets are publicly available for example the BIODID data set*, which is used throughout this thesis. In this review, direct comparisons are made where possible and the advantages/disadvantages of different approaches are discussed.

2.2 Template Based Methods

In this approach a template model of the whole face is built and used to search the image. Face detection is therefore simplified to a 2D pattern recognition problem. This whole face approach is sometimes called “holistic” modelling, see the face detection survey by [42].

Typically a template face model is scanned across a target image, at multiple scales (using an image pyramid [1]) and asked to classify each subregion as face or non-face. Therefore face detection is reduced to a binary pattern classification problem. It is however a highly skewed classification task, because when searching an image

*<http://www.humanscan.de/support/downloads/facedb.php>

very few image subregions actually contain a human face. Several researchers have attempted to solve this pattern recognition problem by modelling the distribution of human faces [89][62][63] or more commonly by applying well known machine learning techniques to discriminate between face/non-face distributions [88][74][65].

One of the earliest face modelling techniques is the “eigenfaces” method developed by Turk and Pentland [89]. This approach models face texture by projecting pixel values from a subwindow containing the face into a linear subspace using Principal Components Analysis (PCA) [44]. The eigenvectors of this space (the “eigenfaces”) represent the main modes of variation learnt from the training set. Two face images can then be compared by projecting into the subspace and recording the “distance in feature space”, which is a Mahalanobis type metric.

The eigenface model can also be used to discriminate between face/non-face image regions, by projecting the region into the subspace and computing the “distance from feature space” measure [89]. This measure computes the residual texture variation of the subwindow, which the face model is unable to model. Therefore subwindows which resemble the training set of face images, will be accepted and non-face like image regions will be rejected.

The eigenface method is primarily used for face recognition rather than face detection. Therefore the primary focus has been face recognition, for which eigenfaces have proved successful [62][63]. In the Sept 1996 FERET face recognition competition [67] the combined face detection/recognition system due to Moghaddam and Pentland [62] achieved a success rate of 95% when applied to a database consisting of 7,562 images of 3,000 people. However in these images the face detection task is reasonably easy (i.e. controlled conditions with uniform backgrounds). There are no detection results reported for later more challenging data sets (see Tables 2.1 and 2.2).

A template method that utilises neural networks to solve the face/non-face classification problem is due to Sung and Poggio [88]. They model the distribution of faces using 6 prototype distributions and also the nearby non-faces using another

6 prototype distributions. The distance of a candidate region from each of these 12 distributions is used to provide a 12 element feature vector. A neural network applied to this feature vector classifies the region as face/non-face.

The face distributions are learnt by applying a modified k-means clustering algorithm to a set of 1067 face patterns (19*19 pixels). Each face pattern is preprocessed by fitting a linear function to the window to correct for lighting variation and further normalised by histogram equalisation [85]. The non-face distributions are learnt by applying clustering to a set of non-face images, which have been normalised using the same preprocessing steps. An eigenspace model is built for each of the 12 clusters. For each training example (face + non-face), a 24 ($2*12$) element feature vector is computed by calculating the Mahalanobis distance and Euclidean distance from the centre of each cluster. A Multi-Layer Perceptron (MLP) is trained to distinguish faces and non-faces based on this 24 element vector.

A large number of face images are collected to form the faces example set (1067 images). However, the space of non-faces is much larger and impossible to represent with a large set of random images. Therefore Sung and Poggio employ a bootstrapping method, originally described in [87], to incrementally add false positives from previous detectors to the non-faces data set (see Algorithm 2.1). When the number of false matches is sufficiently low the boot-strap (Algorithm 2.1) terminates and the final version of the detector is retained. This boot-strap method has since been utilised by other researchers [73] [74] [80].

Algorithm 2.1 Boot-strap Training [87]

1. Start with a set of face images and a small set of non-face images.
 2. Train the detector with the face images and current set of non-face images.
 3. Apply the detector to a set of images not containing any faces and collect all false matches.
 4. Select a subset of the false matches and add them to the training set. Go to step 2.
-

To test their face detector Sung and Poggio collected a set of images containing human faces and background clutter, from various sources e.g. scanned photographs from newspapers and images from the web. This test set contains 136 faces amongst 23 images and has since been used by various researchers to compare face detection algorithms (see Table 2.1) and is known as the MIT data set[†]. The Sung-Poggio detector is able to find 79.9% of the faces with 5 false positives on the MIT images, see Table 2.1.

Method	Number of False Detections						
	0	3	5	8	13	20	42
Sung-Poggio [88]	-	-	79.9%	-	-	-	-
Rowley [74]	74.8%	-	-	84.5%	-	-	90.3%
Viola-Jones [92]	-	-	77.8%	-	-	-	-
Osuna [65]	-	-	-	-	-	74.2%	-
Schneidermann [80]	-	-	79.7%	-	84.6%	-	-
Roth [73]	-	94.1%	-	-	-	-	-

Table 2.1: Comparison of detection rate versus number of false detections for various algorithms applied to the MIT data set (subset B of the CMU data set).

Osuna *et al.* [65] use a similar scheme to that of Sung and Poggio. They subsample 19*19 pixel regions of the face, apply lighting correction and histogram normalisation. However, Osuna *et al.* train a support vector machine (SVM) to discriminate between face/non-face regions instead of a neural network. The SVM [90] learns a decision boundary that can be used directly to classify a 19*19 region. Osuna *et al.* also use the bootstrapping technique of (Algorithm 2.1) to create a representative non-face training set by building the face detector multiple times and collecting false positives from a set of images not containing faces.

The final detector is applied to the MIT data set. The SVM method produces a detection rate of 74.2% and 20 false detections, so is comparable to the results obtained by Sung and Poggio (see Table 2.1). The authors state that the SVM classifier is 30 times faster than that of Sung and Poggio, due to the small number

[†]The MIT test set was later combined with data from Carnegie Mellon University and now forms subset B of the CMU data set described in Section 3.6.3 of this thesis.

of support vectors that need to be evaluated for each image region. However, the method is still slowed down by the requirement to perform lighting correction and histogram normalisation to every subwindow of the image that is examined.

Rowley *et al.* [74] trained a neural network directly on the image pixels, to partition the image subwindows into face/non-face regions. This method models the face as a 20*20 pixel template. Again each extracted subwindow is pre-processed using lighting correction and histogram equalisation. A Multi-Layer Perceptron (MLP) is then applied to the normalised subwindow to classify the region as a face or non-face. The system is summarised in Figure 2.1.

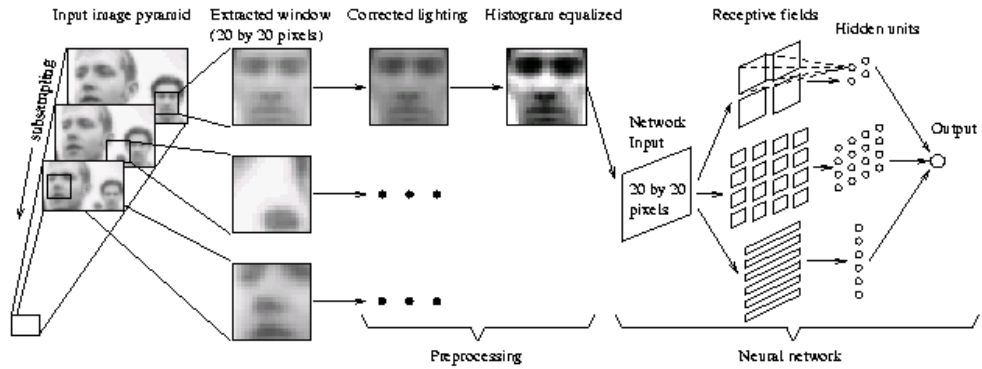


Figure 2.1: System diagram for the Rowley Detector, reproduced with permission from [74]

Several face detectors are built by Rowley *et al.* using different MLP structures. The method is then refined by combining the results of the separate face detectors. Simple voting or AND/OR logic operators are used to improve performance. On the MIT data set, the Rowley-Baluja-Kanade detector achieves 74.8%/0 false positives or 84.5%/8 false positives (see Table 2.1), so has similar performance to the Sung-Poggio detector. Rowley later expanded the original MIT data set (136 faces/23 images) to form the CMU data set (507 faces/130 images). Both the MIT data set and CMU set have since been widely used by researchers to compare face detection algorithms.

On the CMU data set the Rowley Detector is able to detect 89.2% of the faces with 95

false positives over the whole test set [75]. The detection rates for different numbers of false positives are shown in Table 2.2. Table 2.2 shows that the Rowley Detector is comparable with other later detection methods.

Method	Number of False Detections							
	10	31	65	78	95	120	167	422
Viola-Jones [92]	78.3%	85.2%	89.8%	90.1%	90.8%	-	91.8%	93.7%
Rowley [74]	83.2%	86.0%	-	-	89.2%	-	90.1%	89.9%
Schneidermann [80]	-	-	94.4%	-	-	-	-	-
Roth [73]	-	-	-	(94.8%)	-	-	-	-
Fröba [28]	-	-	-	-	-	87.8%	-	-
Fröba [29]	89.0%	90.0%	-	-	-	-	-	-

Table 2.2: Comparison of detection rate versus number of false detections for various algorithms applied to the CMU data set, (expanded version of table appearing in [93]).

A successful method is described by Roth *et al.* [73], using the sparse network of winnows (SNoW) architecture [72] to discriminate between face and non-face regions. They use a 20*20 pixel region and a training set of 1681 face examples. Again all images are corrected for lighting variation and histogram equalisation applied. The bootstrap training method is also used (see Algorithm 2.1).

The method produces a detection rate of 94.1% with only 3 false positives on the MIT test set and 94.8% with 78 false detections on the CMU test set, see Table 2.2, so is a highly successful method. However, the CMU result excludes 5 images containing hand drawn faces, so presumably would give a lower detection rate if tested on the full CMU test set.

The Roth Detector is also slow, because like the Rowley-Baluja-Kanade detector, lighting variation and histogram equalisation must be applied to all subwindows. The SNoW classifier itself is reasonably efficient as it only retains a relatively small number of features from the large number available.

Schneidermann and Kanade [80] model the face patch using an approach based on the naive Bayes classifier. Many classifiers (assumed to be independent) are combined.

Each classifier estimates the joint probability of local appearance and position of a facial subregion.

The method is very slow, as all classifiers need to be applied to every subwindow, but gives good results. On the CMU data set the detection rate is 94.4% with 65 false positives [81], see Table 2.2. On the MIT data set it is 79.7% with 5 false positives, see Table 2.1. Schneidermann and Kanade later extend the method to use a wavelet decomposition instead of facial subregions [81] and demonstrate that the method can also be applied to the detect out-of-plane profile faces (see Section 2.3).

All the template methods discussed so far give good detection results and few false positives, however all of them are very slow requiring at least a few seconds to analyse even a small (300*200 pixel) image on a modern computer. However, Viola and Jones [92] have recently developed a template based method that gives a reasonable detection rate (see Table 2.1 and Table 2.2), but at greatly reduced computational cost. The Viola and Jones “Boosted Cascade Detector” is capable of searching a 384*288 image at 15 frames per second using a 700Mhz PentiumIII processor. This is a vast improvement in speed compared to previous face detectors, even taking into account improvements in computer hardware.

The Boosted Cascade Detector derives its speed from the use of an image representation known as the “integral image”. This structure allows the sum of pixel values in rectangular image regions to be computed quickly, independent of the region size. The need for lighting correction is also removed by computing a second integral image of squared pixel intensities that can be used to variance normalise the feature responses directly.

Given a candidate region the Boosted Cascade Detector algorithm applies several small classifiers, which compute the sum of pixel values from adjacent regions. The output of the classifiers can be combined using a process called “boosting”, which combines many independent, but weak classifiers (maybe only just better than random), into one strong classifier. For the Boosted Cascade Detector a boosting algo-

rithm known as AdaBoost [26], is used to both select individual weak classifiers and weight them appropriately in order to distinguish between face and non-face regions.

The Boosted Cascade Detector has been implemented by the author and is investigated in this thesis. The Viola-Jones method is describe in more detail in Section 3.5. In Chapter 3, the Boosted Cascade Detector is compared with three other efficient face detection methods. The different formulations of the detector and effects of varying training set size are demonstrated in Chapter 4.

Another efficient method is due to Fröba and Küllbeck [30]. This algorithm computes an orientation map of the image at multiple scales. Each scale is then searched using an orientation map template of the face. Fröba and Küllbeck report that their implementation searches a 190x140 image in 80ms using a PII 500Mhz processor. The Orientation Map Detector is also implemented by the author and described in more detail in Section 3.3 of Chapter 3. Later in Section 3.8 it is compared with the Viola-Jones Boosted Cascade Detector [92].

In later work, Fröba [33] extends the Orientation Map Detector by applying the SNoW face classifier devised by Roth *et al.* [73] to verify face candidate regions and therefore reduce the false positive rate of the detector. In conjunction with the SNoW classifier this algorithm is able to achieve a detection rate of 87.8% with 120 false positives on the CMU data set (see Table 2.2).

Fröba applies AdaBoost to the orientation template approach in [32]. This treats each orientation vector as a separate feature and results in improved classification performance. This detector is also extended to detect faces rotated in the image plane [28] (see Section 2.3 for more details).

In a recent work, Fröba [29] uses a modified version of the consensus transform [102] instead of the orientation map. This detector also uses AdaBoost to create a strong classifier template. The consensus transform is shown to be less susceptible to lighting variation, compared to the orientation map. The improved results using consensus

transform are shown in Table 2.2 (Fröba [29]). The detection rate for a small number of false positives using this approach is superior to previous methods, but it is not clear whether the detection rate can be improved further.

2.3 Multi-view Template Methods

The obvious drawback of template based methods, discussed in Section 2.2, is the restriction to frontal upright faces. This section gives a brief overview of extensions to template based search, which aim to detect human faces at multiple view points.

Multi-view face detection is generally broken down into two types of variation; *in-plane* rotation and *out-of-plane* rotation. In-plane rotation refers to cases where the whole face is visible and facing the camera, but the face is not necessarily upright. Out-of-plane rotation is when the face is turned away from the camera, such that some of the facial features are occluded, e.g. a side view of the face.

The huge variety in appearance of the human face, when viewed from different angles means multi-view face detection is much more challenging than upright frontal face detection. However, recently many authors have attempted to extend template methods to solve the problem. The various approaches can be summarised as follows:-

1. One “monolithic” model for all view points
2. Parallel application of multiple face models
3. Pose prediction followed by testing with single model
4. Pyramidal application of multiple face models

The first approach, using one face template to model all possible face views, is widely reported as being unworkable [76][45][97][81][28]. The reason is the highly non-linear

variation of the human head and face when viewed from different angles. No authors suggest building a single template to model all possible views of the head.

Rowley *et al.* [76] suggest using approach two or three for solving the problem of in-plane rotation. Approach two generally involves building multiple detectors for multiple viewpoints and then running all detectors to search for a face. However, in the in-plane rotation case, this can be achieved using a single frontal face detector and rotating the test image. Approach three involves a two stage method, which first tries to predict the most face-like orientation of a given image region and then de-rotates the patch before applying a frontal face detector. In all cases, Rowley’s MLP face template method [74] is used (see Section 2.2 for discussion of MLP approach to upright face detection). A multi-class neural network is trained to perform the pose estimation. The network has 18 outputs, which correspond to 18 in-plane face orientations.

To test detection accuracy of the system, Rowley introduced a test set containing in-plane rotated faces, which is known as “Set D” or the “rotated set” of the CMU frontal face data set[‡]. This image set consists of 50 images, with 223 faces at various orientations. All subjects are looking directly at the camera, but some are upside down or at 90 degrees to the camera. Detection rates and number of false positives when applying Rowley’s method to this test set are shown in Table 2.3.

Method	Number of False Detections				
	45	221	400	600	1345
Rowley [76] (PE)	-	89.2%	-	-	-
Rowley [76] (ALL)	-	-	-	-	96.0%
Jones [45] (PE)	87.0%	89.7%	90.6%	91.6%	-
Jones [45] (ALL)	87.0%	90.5%	92.3%	92.3%	95.0%
Fröba [28]	89.7%	-	-	-	-

Table 2.3: Comparison of detection rate versus number of false detections for various algorithms applied to the CMU in-plane rotations data set.

[‡]http://vasc.ri.cmu.edu/idb/html/face/frontal_images/index.html

Rowley’s approach using pose estimation and de-rotating the face finds 89.2% of faces, with 221 false positives (see Table 2.3, Rowley(PE)). Rotating the test image multiple times and applying Rowley’s upright detector at all orientations results in an improved detection rate of 96.0%, but an increased number of false positives (see Table 2.3, Rowley(ALL)). The two approaches are not therefore directly comparable, because only one point on each ROC curve has been computed for each approach. Whether the pose-estimated method is successful, compared to the run at all orientations approach, depends on the accuracy of the pose estimator relative to the frontal face detector. Similarly the speed of both methods depends on the complexity of the pose estimator relative to the frontal face detector. Rowley *et al.* report that the pose estimated (PE) method is “much quicker” than the *try all* approach [76].

Rowley also suggests an extension of the MLP pose estimator approach to out-of-plane rotation, with 5 separate models - left profile, left half profile, frontal, right half profile and right profile. However, no results are presented for this system [76].

Schneidermann and Kanade [81] use approach three to attempt to solve the out-of-plane rotation problem. They build separate frontal face and right profile detectors using the successful naive Bayes template detection method described in [80] (see Section 2.2). The test image is also reflected, so that left profile faces can also be detected using the right profile detector. The test image is therefore searched three times.

To test the method, Schneidermann [81] introduces a test set for out-of-plane detection algorithms. This test set contains 208 images with 441 faces (347 in profile view). It is publicly available[§] and referred to as the “CMU out-of-plane rotations” set. Results of applying the Schneidermann detectors to this data set are presented in Table 2.4.

The Schneidermann detector finds 75.2% of all faces for 12 false detections rising to 92.7% of faces for 700 false detections. Therefore the method is reasonably successful

[§]<http://vasc.ri.cmu.edu/idb/html/face/profile.images/index.html>

Method	Number of False Detections							
	8	12	34	89	91	221	415	700
Schneidermann [81]	-	75.2%	-	-	85.5%	-	-	92.7%
Jones [45] (PE)	-	-	-	-	-	(75.5%)	-	(83.0%)
Jones [45] (ALL)	-	-	-	-	-	(76.0%)	-	(84.0%)
Wu [97] (PE)	79.4%	-	84.8%	-	-	89.8%	-	-
Wu [97] (All)	-	-	84.1%	86.2%	-	-	91.3%	-

Table 2.4: Comparison of detection rate versus number of false detections for various algorithms applied to the CMU out-of-plane profile data set, (expanded version of table appearing in [97]).

and gives one or two false positives per image, when set to find 90% of faces. The original template method is very slow due to requiring three detectors in parallel. This approach is successful, but very computationally expensive.

Jones and Viola [45] extend their successful upright face detector [92], to cope with in-plane and out-of-plane rotated faces. Similar to Rowley [76], they use a pose estimator to predict the correct orientation model and also compare this approach with the run-all detectors approach. A decision tree classifier, trained using the C4.5 algorithm (Quinlan [70]), is used to predict the face orientation of a given image patch.

When comparing the Jones [45] in-plane rotation approach with Rowley [76] there are some implementation changes, which are required to preserve the speed of the original Viola-Jones approach. For example multiple models are built at 12 orientations, instead of de-rotating the image patch at run-time. Similarly when applying the run-all approach, the multiple models are applied to each image, instead of rotating the test image.

Table 2.3 shows that on the CMU in-plane rotation test set the Jones method gives very similar performance to the Rowley approach. When using the pose estimation (PE), Jones finds 89.7% of faces for 221 false detections, versus 89.2% using Rowley. Similarly when applying all detectors, Jones finds 95.0% of faces for 1345 false detections, versus 96.0% using Rowley. Therefore the main benefit of the Jones approach compared to Rowley is speed. When searching a 320x240 pixel image, the

Jones method requires just 140ms using pose estimation and 660ms using the run-all approach (with a 2.8Ghz Pentium4 Processor).

The Jones [45] approach is also applied to out-of-plane face detection. Here the decision tree is trained to distinguish between left and right profiles. The detector ignores frontal face images, but is still applied to the CMU out-of-plane test set. Table 2.4 shows that similar results are achieved using pose estimation (PE) and running both left/right profile detectors in profile (e.g. 83.0%/84.0% for 700 false matches). This result is not directly comparable with Schneidermann because frontal faces are not considered by Jones, however the Jones detection rate for 700 false matches is still much lower (84.0% vs 92.7%). Therefore the Jones multi-view method is much more efficient, but performs worse than Schneidermann for out-of-plane profile faces.

Y.Li *et al.* [51] also use a pose prediction method to detect multi-view faces. They build models which cover all 8 combinations of up/down, frontal/profile and left/right face views. Each image window is preprocessed using horizontal and vertical Söbel filters. SVM Regression [90] is used to predict the pose angle and select the appropriate face detector [50]. The individual detectors are a hybrid of the eigenface model [62] and the SVM face classifier [65] (see Section 2.2 for a description of these template methods). However these detectors are applied to Söbel filter outputs, not the normalised image pixels.

This approach is more efficient than applying all 8 face detectors to every face patch, but it is still computationally expensive to test an individual image window. Therefore this technique is only used to test small regions of the image, which have been detected using motion or skin colour techniques [57]. The method is shown to give accurate results when used to track faces in video sequences [51], but is not applied to any of the CMU test sets.

S.Li *et al.* [49] describe a multi-view detector based on the Viola-Jones upright face detector. Their method finds out-of-plane rotation faces using an algorithm they describe as the “detector pyramid”, which can be considered a fourth approach to

multi-view face detection. This method first applies one detector to distinguish face and non-face regions (all views from left to right profile are modelled). If a region passes this detector another level of detectors is applied. Each level of detectors restricts the range of face profiles modelled, in a coarse-to-fine manner. If all detectors at a particular level fail then the region is rejected. In-plane rotation is dealt with by applying the detector pyramid to rotated versions of the test image. The pyramid detector is very efficient, so this multi-view method is very quick (200ms per 320x240 image on a PIII 700Mhz PC). S.Li *et al.* apply their algorithm to the CMU out-of-plane profile set, but unfortunately provide no quantitative multi-view results in their paper [49], therefore it is not possible to assess the effectiveness of this approach.

A similar pyramidal scheme is adopted by Fröba [28]. Fröba adapts the boosted orientation map frontal face detector [32] (see Section 2.3) to the task of in-plane rotated face detection. The first detector accepts faces in the range $(-60^\circ, +60^\circ)$ or immediately rejects the image patch. The second level of detectors partition the orientations into finer ranges, or reject the image patch if all detectors fail. The final level applies the SNoW detector due to Roth *et al.* [73] at the appropriate orientation.

Fröba applies this method to the CMU in-plane rotations test set (see Table 2.3) and achieves a reasonable detection rate of 89.7% for only 45 false positives. However, Fröba's method is difficult to compare with other methods because the ROC curve does not extend to higher true/false detection rates. The restriction to faces in the range $(-60^\circ, +60^\circ)$ also puts an upper limit on the detection rate, which can be achieved. However, the decision tree algorithm is very fast (40ms for a 320x240 pixel image using a 1000Mhz Athlon processor).

Wu *et al.* [97] describe a multi-view extension to the Viola-Jones cascaded face detector [92] approach (see Section 2.3). This method is a hybrid of approaches two and four. Face detectors are built for 5 out-of-plane orientations and 3 different in-plane orientations (i.e. 15 detectors in total). In version one of this approach all detectors are run in parallel. In version two, only the first six layers of each (16 level) cascaded

detector are applied. The most promising detector after the first 6 layers is taken as the likely orientation of the face and the remaining layers evaluated. Hence the image patch is ultimately accepted or rejected by one appropriately orientated detector.

The results of applying the Wu method [97] to the CMU out-of-plane test set are shown in Table 2.4. The method outperforms Jones [45] and gives performance similar to Schneidermann [81]. However, the Wu algorithm is preferable to Schneidermann due to the more efficient detectors used, which means the system can approach real-time on a fast machine (80ms for a 320x240 image on a P4 2.4Ghz PC).

The pose estimation (PE) version of the Wu algorithm, gives similar results to the run-all approach (see Table 2.4) and gives a speed up factor of 1.7. Wu *et al.* also apply their algorithm to a test set with simultaneous 360° in-plane and out-of-plane rotated faces (the CMU out-of-plane test set contains only approximately upright non-frontal faces). For this they use 12 in-plane orientations and 5 out-of-plane orientations, hence 60 detectors in total. They give a few example results on their own test set. This simultaneous in-plane/out-of-plane multi-view algorithm requires just 250ms on a 320x240 image (using a Pentium4 2.4 Ghz Processor).

The multi-view detection methods discussed in this section are all extensions of the upright frontal template detection methods described in Section 2.3. However, multi-view face detection is much more challenging than upright frontal face detection, therefore the performance of multi-view algorithms tends to be worse (lower detection rates and more false positives). The most successful and computational efficient approach is probably due to Wu [97], which extends the highly successful Viola-Jones face detector [92]. Approaches which try to predict the correct orientation (e.g. Jones [45] and Rowley [76]) tend to be more brittle and not perform as well.

2.4 Feature Based Methods

An alternative to using a single template to represent the whole face is to search for individual facial features and then declare an image region to be a face if an appropriate combination of facial features are identified. The detection of smaller facial features instead of the whole face makes the detection of not quite frontal faces more robust (e.g. faces with small amounts of in-plane and out-of-plane rotation). However, the face generally needs to be larger in the image, so that internal facial features can be detected. Therefore many feature based methods are unsuitable for detection of low resolution faces, e.g. 30*30 pixels.

For example Leung *et al.* [48] use multi-orientation, multi-scale Gaussian derivative filters to detect facial features. The vector of filter responses are learnt for 5 points on the face, namely both nostrils, both pupils and the nose lip junction. The bank of filters is applied to an image and the best matches (using a dot product metric) are retained as candidate nostrils, eye etc. The facial features are combined into possible face candidates. This is achieved by selecting all feature points above a certain threshold and pairing them up. Each pair is then used to define small elliptical regions where the remaining 3 feature points are likely to occur. If further feature points are found within the predicted regions, then a face candidate is formed.

Given a face candidate the inter-point distances form the basis of a likelihood score, assuming a Gaussian distribution of inter-feature distances computed from the training set. A refinement, implemented by the Burl and Leung [4], is to use the shape statistics of Dryden and Mardia [20], instead of just inter-feature distances. For both methods, the candidate shape with the highest likelihood score is deemed the location of the face.

This feature based method achieves a 95% success rate with approximately frontal face images with cluttered backgrounds [48]. However, results are only provided for a small database that is not publicly available. The feature detectors do have some

tolerance to head rotation (ie out of plane and in plane rotation), except at extreme angles. The method is also able to cope with occlusions, in some cases, because not all 5 feature points need to be found to return a face candidate. Therefore this method is reasonably robust.

The system developed by Yow and Cipolla [99] [100] [101] also uses a Gaussian second derivative filter to detect facial features. In this case the filters are merely elongated horizontally in the ratio 3:1 to match the approximate shape of facial features, such as the eyes and mouth. The feature point candidates are then formed using non-maximal suppression of the Gaussian filter response. The candidate features are then combined to form face candidates. Shape is not modelled explicitly, but implicitly using a grouping method based on belief networks [78]. The quality of the feature response is propagated up the network (using an algorithm due to Pearl [66]) and the highest scoring configuration of points selected as the location of the facial features. This method was able to produce a success rate of 93% on 135 test images [100]. Again the images are not publicly available, so it is difficult to make comparisons between this algorithm and the method due to Leung *et al.* [48].

Cootes *et al.* [11] implement a feature detector based on Eigen-features [61] to approximately locate the face in the image, before applying a local ASM search [13] (see Section 2.6). Four feature detectors are trained to recognise the grey level patterns around both pupils and both nostrils by projecting into a low dimension eigenspace, in a manner similar to the original eigenface approach, due to Turk and Pentland [89]. When searching, each feature detector is scanned across the image (at multiple scales and orientations) and the closest matches (calculated by proximity in each feature eigenspace) are declared candidate points for the pupils/nostrils etc.

The candidate points are then grouped into face candidates, which are tested by fitting a shape model [8] [20] to the (possibly incomplete) set of facial features. The face candidate with the most number of feature points and best match to the shape model is declared the location of the face. This approximate location and scale is

then used to seed a local active shape model (ASM) search [13].

The system was able to find 35 faces in a test set of 40 [11], giving an overall success rate of 87.5%. Here a success is declared when the top scoring face candidate returns a starting point for the ASM algorithm, which is close enough to the true face position for the ASM to converge on the face. However, testing is only performed on a small number of images. The method is quite slow, taking 12 seconds on a SUNSPARC20 computer, but would be much faster on a modern computer. An algorithm similar to this one is evaluated as part of this thesis, see Section 3.2 of Chapter 3.

A more recent feature based approach is due to Hamouz *et al.* [39], which utilises a bank of Gabor filters to search for 10 facial features (eye corners, eye centres, nostrils and mouth corners). Each feature is modelled using a Gaussian Mixture Model (GMM) of feature responses. Any triplet of feature detections, with an acceptable spatial orientation, produce a face location hypothesis. These face candidates are then normalised using an affine transformation and tested using an SVM region classifier, similar to the one used by Osuna *et al.* [65] for template based face detection (see Section 2.2). The highest ranking candidate based on the SVM discriminant function is declared the location of the face.

One advantage of using Gabor filters and searching over all scales and orientations is that the implied scale and orientation of each detector can be used to limit the search for feature triplets. This is achieved by computing limit regions for the relative distribution of individual features [37]. Otherwise the number of possible triplets becomes prohibitive, if too many candidate locations are returned by each feature detector, resulting in a combinatorial explosion. However, applying 10 feature detectors to the whole image is computationally expensive. The reported speed of the whole method is 13 seconds per image (however the implementation is not optimised for speed and could be much quicker [39]).

Hamouz *et al.* apply their algorithm to the XM2VTS and BIODID test sets (see Section 3.8 for a description of these two data sets). The algorithm finds 91% of eye

locations in the XM2VTS within 10% of the true inter-ocular eye separation. The success rate on the BIOD data set is 65%. This is similar performance to the Jesorsky *et al.* [43] localisation method, discussed in Section 2.5. The methods described in this thesis are compared with the Hamouz and Jesorsky algorithms in Section 7.7.

Graf *et al.* [35] applied band pass filtering and detect facial features using morphological operations, which are then grouped to form face candidates. Maio and Maltoni use the Söbel filter to extract edge orientation features. The generalised Hough transform [2] is then used to detect the elliptical outline of the face. All these methods report good results, but on different data sets, so are difficult to compare.

A general weakness of all feature based methods is that they depend on the accurate detection of certain set of features. If this initial search fails, then the face will not be detected. Individual feature detectors are often noisy, due to the limited local image structure around a given feature compared to the richer structure present in the whole face. Some feature based methods have in built redundancy, to missing features e.g. [4][11][54][43][100]. However, if the face is at very low resolution (e.g. 30*30 pixels) none of these methods will be successful, because it is not possible to discern individual features.

Another limitation of feature based face detection is that if the feature detectors generate too many false matches then there is a combinatorial explosion in the number of possible face candidates. This can make feature based methods slower than whole face methods, even if the individual feature detectors are simpler and faster than the whole face detector.

A problem with comparing feature based algorithms is that no common test set has been adopted by researchers, therefore it is difficult to make performance comparisons. The CMU data set, used to compare template methods in Section 2.2 contains many low resolution faces, so is not suitable.

2.5 Feature Localisation Methods

This section reviews previous methods for facial feature finding. The methods discussed attempt to find facial features such as eyes, nose and mouth given an approximate location of the face. These algorithms assume that the face has been found correctly and facial features are present in the image, although they may be occluded by glasses, hands etc. Many methods are similar to those discussed in Section 2.4. However, in Section 2.4 the configuration of candidate features is used to discriminate between faces/ non-faces. Here the purpose of feature detection is merely to locate a given set of facial features within a predefined target region.

A common feature detection approach is to extend a whole face detection method, to search for smaller facial features at a higher resolution. For example Feris *et al.* [23] use Gabor Wavelet Networks(GWNs) to first find the approximate face region and then use smaller GWNs to look for 8 individual features, namely the corners of the eyes, the nostrils and mouth corners.

Gabor Wavelet Networks [46] consist of a weighted set of wavelets, each having an associated position, orientation and scale within a common co-ordinate system. A GWN is trained by randomly dropping wavelets onto an image and optimising the parameters to minimise the difference between the training image and the GWN representation. Each GWN therefore compresses the data present in the training image, see Figure 2.2 for examples of GWN compression.

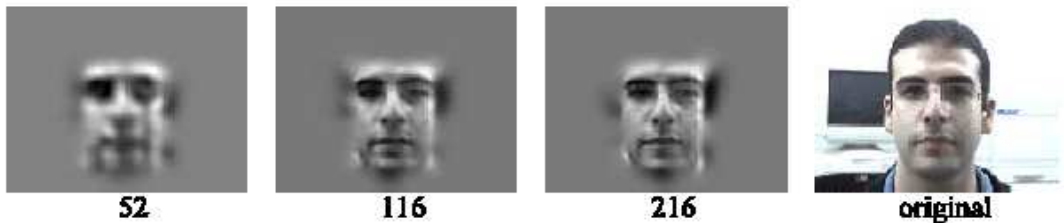


Figure 2.2: Example of image compression using various numbers of Gabor Wavelets, reproduced with permission from [23]

A GWN is fitted to an unseen image, by finding the linear transformation from the model frame that enables the GWN to best reconstruct a portion of the image. The match can be computed efficiently without reproducing the image pixel values explicitly [23]. The first stage of feature localisation is to fit a whole face GWN to the image, which suggests approximate locations for each feature. The second stage is the refinement of the approximate feature prediction using GWNs trained on each feature. The initial whole face GWN is influenced by all facial features and therefore less accurate than the local GWN feature search.

Feris [23] states that each feature is found within 3 pixels of the landmarks points in $\sim 95\%$ of cases for eye corners and nostrils and in $\sim 88\%$ of cases for mouth corners. The test set is a subset of the Feret database [67], where the faces have an approximate inter-ocular separation of 50-60 pixels. Here, 3 pixels represents 5-6% of the inter-ocular separation, so the GWN method is accurate, but not always reliable and is only tested on a relatively clean data set. The method is also rather slow, so is unsuitable for real-time applications.

Jesorsky *et al.* [43] use the Söbel filter to detect edges in the image and then match strong edges to a face edge model using the Hausdorff distance [77]. This initial face localisation method is then extended by searching with a smaller model to refine the eye locations, as shown in Figure 2.3. The location of each pupil is further refined using a MLP based detector for each eye.

Jesorsky *et al.* introduce the BIOID data set, which is described in Section 3.6.2 and used throughout this thesis. This three stage method is able to find the centre of the eye pupils within 10% of the inter-ocular distance for 80% of the BIOID database (described in Section 3.6.2) and 92% of the XM2VTS database (discussed in Section 3.6.1). The Hausdorff matching is quick, requiring just 30ms for both coarse matching and refinement on a PIII 850Mhz processor, but the speed of the final MLP eye detector stage is not commented on. In Section 7.7, the methods presented in this thesis are shown to give similar results to the Hausdorff+MLP search on the

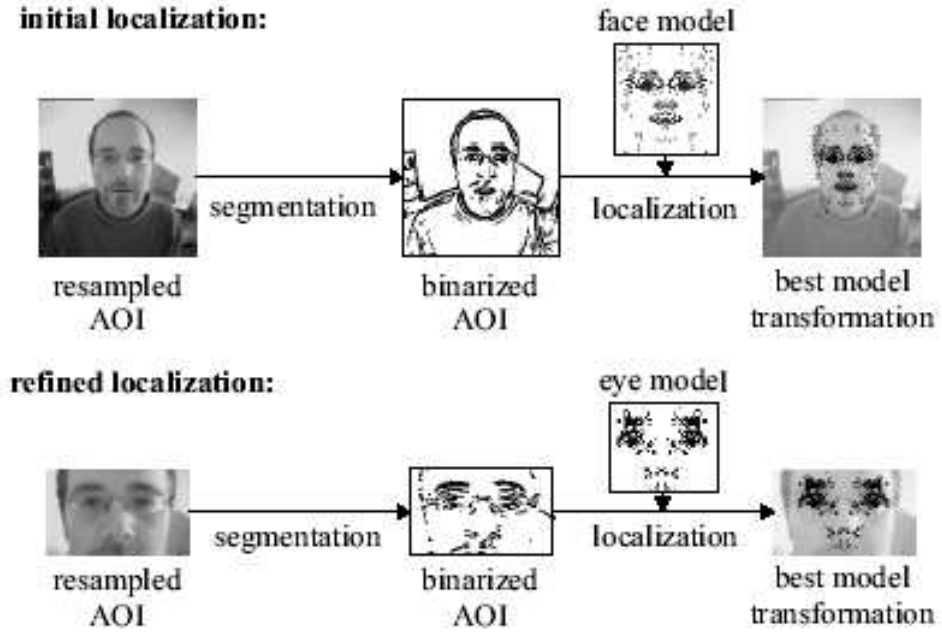


Figure 2.3: Overview of face localisation using the Hausdorff distance, reproduced with permission from [43]

XM2VTS data set, but outperform the Hausdorff+MLP method on the BIOID data set.

Reinders *et al.* [71] describe a system to detect the right eye using neural networks and orientation maps (see Section 3.3.1). Four eye features are detected, namely the inner and outer corners of the eye and the upper and lower eyelids. The configuration of strong responses from each of the 4 neural networks is used to determine the likelihood of a region being the true location of the right eye. The system detects 96% of right eyes within 2 pixels, on faces with an inter-ocular separation of 20 ± 2 pixels. However, a good initialisation is required, for the eye finder. In fact, Reinders *et al.* use video and initialise the search from the previous frame, which limits the search region for the eye detector. It is unclear how well this system would work if the eye search was less constrained.

Herpers *et al.* [41] use steerable filters [25] to find strong edges on the face. They use high resolution face images (512*512 pixels) and recognise key points on the face,

e.g. eye and mouth corners using hand built heuristics. For example a point on an edge of high curvature close to a spherical edge is assumed to be the corner of the eye and the spherical edge is the outline of the pupil. This approach achieves a success rate of 97.3% with the iris, 94.4% with eye corners and 83.8% with mouth corners. However, the algorithm is only tested on a private data set, that does not contain any occlusion (e.g. due to facial hair or glasses) and all images are collected under controlled lighting conditions. Therefore the method is unlikely to work in realistic situations and is only usable on high resolution faces.

Shakunaga *et al.* [83] describe a system which combines the Eigen-template approach of Turk and Pentland [89] and a 3D model of the human face. Eight facial features are used; namely the eyes, eye brows, ears, mouth and nose. A common Eigen-space is built for each feature with a training set of left, right and frontal views. These detectors scan the image to produce a set of feature candidates that also indicate a direction (i.e. left/right/frontal). The feature candidates are combined into groups to form face candidates, subject to geometric constraints. The approximate pose of the face is estimated and a 3D shape model used to predict the location of missing features. A refined search for missing features then takes place. The final evaluation function is a weighted combination of the sum of feature responses, the fit with the 3D shape model and a penalty for missing features.

Shakunaga report finding $\sim 98\%$ of the internal facial features and $\sim 87\%$ of the ears testing on frontal face images and a few percent worse results using a test set with the subject at 30° to the camera. However, they do not specify the accuracy required to define a correctly detected feature and only test on their own data. They also appear to only test performance at discrete angles i.e. $-30/0/+30$ degrees and do not comment on intermediate angles. The speed of the system is not commented upon.

2.6 Iterative Search Methods

Iterative search methods project a model into the image and update the model parameters given the fit of the model to the nearby image pixels. An example is the approach due to Wiskott *et al.*[95] known as “Elastic Bunch Graph Matching”. Here a “Bunch Graph” consists of a set of “Gabor Jets” and a graph structure. A Gabor Jet is a vector of responses to a family of Gabor filters and is used to model each facial feature. Forty Gabor filters are used at 5 frequencies at each of 8 orientations. The graph structure models the distribution of distances between adjacent features.

The first stage of this algorithm is to find the approximate face location by searching with a rigid graph and finding the best match to a set of averaged Gabor jets. The second stage is to perform an iterative search of the face region to refine the position and size of the face. At this stage the full set of Gabor Jets is used. The disparity between each Gabor Jet from the Bunch Graph and the Gabor Jet computed from the image is used to predict small local displacements to improve the location of each feature. The third stage is the same as stage two, but allows variation in aspect ratio. Finally a small refinement of each Gabor Jet takes place, weighted by the distortion of the inter-feature distances relative to the graph model.

The Wiskott method is primarily a face recognition algorithm, as the final Gabor Jet responses are used to identify individuals. Therefore the accuracy of the feature localisation is not discussed in detail. However the system is quoted as giving an average point to point error of 1.6 pixels [95], for faces with an inter-ocular distance of ~ 40 pixels. The system is therefore accurate to within $\sim 4\%$ of the inter-ocular separation, but is relatively slow, requiring 30 seconds on a SUNSPARC 10-512.

Gabor jet features have also been utilised for the task of face feature tracking [56] [58] [94]. For example Maurer *et al.* [56] compute the phase disparity between consecutive frames to predict the direction of movement of individual feature points. Maurer *et al.* initially track multiple feature points through a sequence of frames and then

reverse the process to select a set of reliable features for each individual. The bunch graph approach (described above) is then used to constrain the relative movement of features in future tracking.

McKenna *et al.* [58] also use phase disparity to track features, but constrain the detection using a statistical shape model (which is further described below and discussed in detail in Chapter 5 of this thesis). A confidence measure is assigned to each feature point, based on the difference in Gabor Jet representations between frames. This allows the shape model to be fitted to the predicted feature points in a weighted manner, which allows unreliable matches to be excluded and avoids the final set of predicted points forming an unlikely configuration.

Wiegardt *et al.* [94] adopt the shape constraint method developed by McKenna, but compare it with a novel approach which encodes the shape constraints directly into the Gabor Jet phase-based disparity motion estimation. Wiegardt *et al.* claim improved point to point error when applying their shape constraint technique to a series of frames. However all these tracking extensions to the original Wiskott method [95] are difficult to compare due to the lack of a common test set.

Another method that combines shape constraints and feature detection is the active shape model (ASM) due to Cootes *et al.* [13]. The shape is learnt from a set of manually landmarked images using the shape statistics of Dryden and Mardia [20]. Building such a shape model is described in detail in Chapter 5, but can be summarised as aligning the shapes to a common co-ordinate frame and performing a principal components analysis (PCA) on the aligned data. A profile model is trained on a linear patch passing through each feature point within the shape model. The set of profile models and shape model form the ASM.

Given a sufficiently accurate starting position the ASM search proceeds by searching along each linear profile to improve the local match. The best match for each profile is then recorded and the shape model fitted to the new shape. Constraints are placed on the shape model, so that spurious matches by an individual profile model are

ignored. The search then proceeds iteratively with a new search along each profile. The ASM uses an image pyramid [1] to perform a coarse-to-fine search.

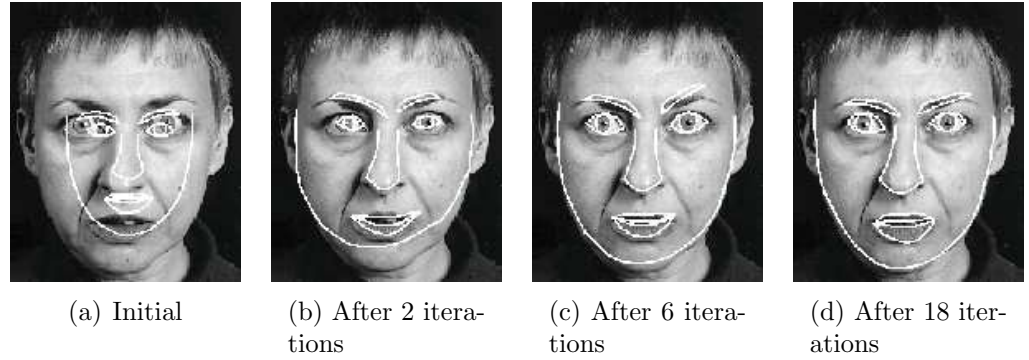


Figure 2.4: Examples of valid and invalid feature detections

Good results fitting ASMs to unseen faces are reported in [7]. The ASM is trained on a set of 200 face images with 133 landmark points and applied to a test set of 200 unseen faces with equivalent manually labelled points. The ASM was initialised with the mean shape and displaced from the true location of each test face by ± 10 pixels. The resulting search gave an average point to point error of 4.8 pixels, with 1.0% of searches failing to match to the face (and therefore not included in the mean error calculation). Relative to the inter-ocular separation of 70-80 pixels, this represents a point to point error of $\sim 6.2\%$. This is a good result considering the large number of feature points evaluated, many of which are implied points, e.g. equally spaced points along the outline of the face.

A later alternative to the ASM is an approach known as the Active Appearance Model (AAM) [6]. The AAM is described in detail in Chapter 7. Like the ASM, the AAM is projected into the image and searches iteratively for the best match using a coarse-to-fine search. However, instead of using profile models to search around each feature point, the AAM uses the difference between the texture model and the image to drive the shape and texture parameters.

The ASM and AAM algorithms are compared by Cootes *et al.* [7]. The AAM is found to give a point to point error of 4.0 pixels with 1.6% failures, compared to 4.8

pixels using the ASM with 1.0% failure rate, when both are applied to the same test set. The AAM uses all the image data, instead of just profile texture, which makes the search more robust and locally accurate. A practical advantage of the AAM approach compared to the ASM is that a successful model can be built with fewer control points (i.e. less than 133), because the texture model is able to capture some of the shape variation, whilst the ASM requires many control points to adequately model the local texture variation using profiles.

A weakness of both the AAM and ASM methods is that they require a good starting position to converge to the correct solution. The AAM must start within approximately ± 15 pixels of the correct location for a face of width ~ 200 pixels [6], with approximately correct orientation and scale. The ASM has a slightly larger capture range, compared to the AAM [7], however if either is initialised too far away from the face it will find a false minima and fail to find the correct solution. These methods are therefore only suitable for local search. Different variations of the AAM search are evaluated in Chapter 7 of this thesis.

The AAM algorithm has also been extended to face tracking applications. For example Dornaika and Ahlberg [19] extend the AAM to a 3D wireframe model of the human face, with a separate texture model. They first compute the 3D head pose using a RANSAC (RANDOM SAMPLING CONSENSUS [24]) technique combined with a goodness of fit estimate for the face texture. The AAM algorithm is then used to refine the model fit. This method shows promising results, but is only evaluated (for pose estimation accuracy) on four video sequences.

2.7 Summary

The main conclusion of this review of face and feature detection is that is very difficult to compare algorithms due to the lack of common test sets. This is especially true for feature based face detection (see Section 2.4) and feature localisation methods (see

Section 2.5).

For template based face detection (see Section 2.2) and multi-view face detection (see Section 2.3) the CMU data sets provide common data to compare results. Therefore whole face template methods are more thoroughly tested and appear to provide more robust results, compared to feature based detection, especially for low resolution upright faces. Therefore the main approach to face detection in this thesis is template based.

An important concern for practical applications of face detection is the time required to locate a face. The algorithm must find the face in less than one second and preferably less than ~ 200 ms. With the processor speed of modern computers this is now feasible. However, only certain algorithms are suitable. The Boosted Cascade Detector [92] and Orientation Map Detector [30] are selected and tested in the next chapter along with two other face detection methods.

The feature localisation methods discussed in Section 2.5 can be applied after the face detection stage. This thesis describes three novel methods of combining feature detection and shape modelling (see Chapter 6). These shape constrained methods are compared and eventually combined with the iterative AAM search (see Chapter 7).

Chapter 3

Face Detection Methods

Four algorithms are investigated in more detail. Namely the Linear Profile Detector, Orientation Map Detector [31], Normalised Correlation Detector and Boosted Cascade Detector [92]. The Linear Profile Detector is a feature based method, similar to the method described by Cootes *et al.* [9]. The Orientation Map Detector, Normalised Correlation Detector and Boosted Cascade Detector are template based methods.

3.1 Face Detection Training Set

The WEBCAM image set is used to train a variety of face detectors. The data set consists of 1055 images, containing one face each, with 128 identities, obtained using a web camera in our lab. The WEBCAM images are manually labelled with 20 corresponding feature points, see Figure 3.1.

These feature points allow each face to be cropped and scaled when building template based face detectors. The labelled feature points are used to build individual feature models for the Linear Profile Detector described in Section 3.2 and later to train individual feature detectors in Chapter 6. The WEBCAM training set is used as the

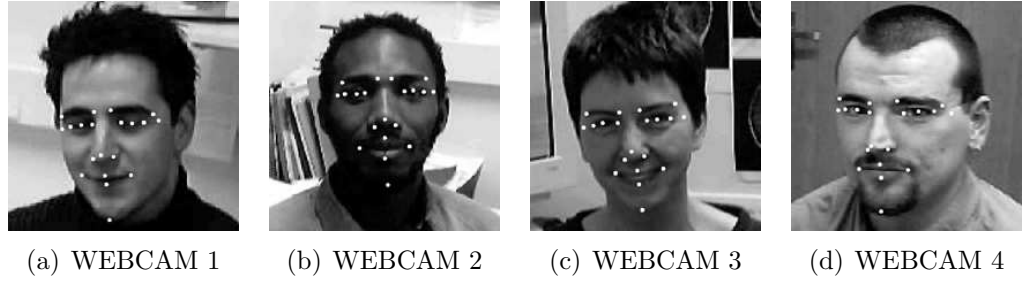


Figure 3.1: Example WEBCAM training images with 20 landmark points

main training set throughout this thesis. The only model to be built using a different training set is the Boosted Cascade Detector (see Section 3.5), which uses ~ 6000 small face images of which the WEBCAM images form a subset (see Appendix C).

3.2 Linear Profile Detector (LPD)

The Linear Profile Detector (LPD) was devised by Tim Cootes at Manchester University and investigated by the author. The LPD is similar in spirit to the feature based methods discussed in Section 2.4. Feature points are modelled using Linear Profile Models (see Section 3.2.1) and a face candidate detected if feature points can be detected in a face like configuration.

3.2.1 Linear Profile Models

A Linear Profile Model characterises a given facial feature point by learning the average normalised image gradient vector in a specified direction. For example a horizontal or vertical line through the eye pupil. For a profile containing n pixels $\{x_i\}$, the gradient vector is

$$\mathbf{v} = (v_1, v_2 \dots v_{n-1}) \quad (3.1)$$

Where $v_i = x_i - x_{i+1}$ is the difference between adjacent pixel values. Additionally the normalised gradients v'_i are computed as follows.

$$v'_i = \frac{v_i}{|v_i| + \mu} \quad (3.2)$$

Where μ is the mean value of $|v_i|$ over all profiles collected from a given face.

Given a training set of face images the mean normalised gradient vector $\mathbf{v}'_{\mathbf{m}}$ is computed. Then the profile match score m_s between a normalised gradient vector \mathbf{x}' from an unseen image and the mean normalised gradient vector $\mathbf{v}'_{\mathbf{m}}$ is computed using the dot product, as follows.

$$m_s = \mathbf{v}'_{\mathbf{m}} \bullet \mathbf{x}' \quad (3.3)$$

The profile model can search along multiple positions along a profile sampled from an image in order to determine the best local match. This form of linear search is very efficient.

3.2.2 LPD Training

The full Linear Profile Detector is trained from the WEBCAM data set (see Section 3.1) by specifying a set of Linear Profile Models, passing through a given set of feature points. In this implementation there are 18 profiles - 16 vertical profiles through each point shown in Figure 3.1 (excluding the temples, chin and nose tip) and two horizontal profiles through the centre of the eyes. Each profile has a manually defined range (p_{min}, p_{max}) where the values of p_{min} and p_{max} are relative to a model frame with the right eye at $(-0.5, 0)$ and the left eye at $(+0.5, 0)$. Given the list of profiles and a set of labelled images, the LPD is built as described in Algorithm 3.1.

Algorithm 3.1 Linear Profile Model Training Method

1. Normalise all shapes to a frame with the right eye at position $(-0.5, 0)$ and the left eye at position $(+0.5, 0)$
 2. Calculate the mean shape
 3. Calculate the range of displacements (r_{min}, r_{max}) over the training set from the mean of each landmark point.
 4. Project each profile into each training image and calculate the image gradient vector $\mathbf{v} = (v_1, v_2 \dots v_{n-1})$
 5. Normalise the gradient vector \mathbf{v} by applying $v_i = \frac{v_i}{|v_i| + \mu}$ where μ is the mean value of $|v_i|$ over all profiles.
 6. For each profile, compute the mean vector \mathbf{v}_m from the set of gradient vectors \mathbf{v} .
-

The set of Linear Profile Models, mean shape and range of displacements for each linear profile define the LPD. The next section describes how to use such a model to search unseen images for faces.

3.2.3 LPD Search

A summary of the LPD search algorithm is as follows:-

1. Search for troughs in the image to provide candidate eye locations
2. Pair eye candidates to form candidate face regions
3. Test each face candidate using Linear Profile Models and return the candidate with the highest score.

Candidate eye locations are detected by building a gaussian image pyramid [1] of the original image and searching each level of the pyramid for troughs - pixels with an intensity lower than their immediate neighbours. Only troughs of a certain depth t_d are accepted, where t_d is defined as the difference in intensity between the centre of the trough and the maximum intensity in the surrounding neighbourhood. This form

of feature detection is very simplistic, but usually finds the eye regions, if the face is reasonably large in the image. However many false matches are also produced.

The next stage is combining candidate eye locations to form candidate pairs. Given n candidate features in a pyramid level, the number of possible pairs is $n(n-1)/2$. To make the face search more efficient, restrictions are placed on the pairing of features, as follows:-

- The angle θ between two features relative to the horizontal must be with the range $(\theta_{min}, \theta_{max})$
- The separation between the two features must be within the range (d_{min}, d_{max}) where

$$\begin{aligned} d_{min} &= \max \left\{ \begin{array}{l} \frac{s}{\sqrt{2}} * 2^L \\ w_{min} * n_x \end{array} \right. \\ d_{max} &= \min \left\{ \begin{array}{l} s * \sqrt{2} * 2^L \\ w_{max} * n_x \end{array} \right. \end{aligned} \quad (3.4)$$

Here w_{min} and w_{max} are parameters controlling the maximum and minimum size of the face relative to the image width n_x . L is the pyramid level and s is the preferred pixel separation.

The preferred pixel separation s is the pixel separation at which a pair of troughs are likely to represent eyes and determines the pyramid level L at which a face is candidate is likely to be formed. The value of s also determines which levels are searched. For example, if $d_{min} > d_{max}$ for a given level L then this pyramid level is not searched - because the constraints cannot be satisfied at this level. Suitable values for the parameters are as follows:-

$$s=10 \quad \theta_{min} = -30^\circ \quad \theta_{max} = 30^\circ \quad w_{min} = 0.1 \quad w_{max} = 0.5 \quad (3.5)$$

The third step is to evaluate all candidate pairs and select the candidate that most closely resembles a face. The algorithm for testing a pair of candidate eye locations is as follows (see Algorithm 3.2).

Algorithm 3.2 Testing Candidate Pairs

1. Given a candidate pair, select the appropriate pyramid level specified by preferred pixel separation s and project all profiles into the image plane.
 2. For each Linear Profile Model k , sample a normalised gradient profile \mathbf{x}'_k in the range $(p_{min} + r_{min}, p_{max} + r_{max})$.
 3. Search along \mathbf{x}'_k for the best match m_s of the mean linear profile \mathbf{v}'_m , as specified by Equation 3.3.
 4. Overall candidate face score $c_s = \sum_k (m_s)_k$
-

The final output of the LPD is the candidate with the highest face score c_s . Note the LPD tends to be used for localisation i.e. it is assumed that only one face is present in the image being searched. It is possible to detect multiple faces, by thresholding the match quality c_s , but this approach is not generally adopted.

3.3 Orientation Map Detector (OMD)

The Orientation Map Detector [30] is a template method that runs a face model over the image to detect upright faces, based on orientation maps (first introduced by Granlund [36]).

3.3.1 Orientation Maps

An orientation map is an image representation which computes a direction and edge strength for each image pixel. Given an image pixel $i(x, y)$, the direction $\theta(x, y)$ and edge strength $s(x, y)$ are computed as follows.

$$g_x(x, y) = \begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix} \star i(x, y) \quad (3.6)$$

$$g_y(x, y) = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} \star i(x, y) \quad (3.7)$$

$$\theta(x, y) = \arctan\left(\frac{g_y(x, y)}{g_x(x, y)}\right) \quad (3.8)$$

$$s(x, y) = \sqrt{g_x^2(x, y) + g_y^2(x, y)} \quad (3.9)$$

Here the horizontal and vertical edge gradients $g_x(x, y)$ and $g_y(x, y)$ are computed using Söbel filter masks [85]. The edge strengths $s(x, y)$ and orientations $\theta(x, y)$ form an orientation vector $v(x, y)$ at each pixel of the orientation map. Orientation maps are useful for object recognition because they ignore some superfluous image properties. For example if a linear transformation is applied to the original pixel values (e.g. addition of a constant or rescaling by a constant) then the orientation map representation of the image is unchanged.

3.3.2 Comparing two Orientation Maps

Orientation maps of the same size are compared by summing the difference between orientation vectors $v(x, y)$ at corresponding (x,y) locations. The distance metric between any two orientation vectors $v_1 = (s_1, \theta_1)$ and $v_2 = (s_2, \theta_2)$ is shown in Equation 3.10.

$$d(v_1, v_2) = \begin{cases} \sin(\theta_1 - \theta_2) & \text{if } |s_1|, |s_2| > s_t \\ 1 & \text{otherwise} \end{cases} \quad (3.10)$$

Hence the distance measure $d(v_1, v_2)$ between two orientation vectors is the sine of the angle between the two vectors, but is unit distance if the edge strength of either orientation vector falls below a threshold. Note, using this distance measure, the polarities of edges in the original image are ignored, as orientation vectors that differ by 180° are considered equal.

3.3.3 OMD Training

An orientation map of the face is constructed from the WEBCAM training set described in Section 3.1. The training images are aligned using the manually labelled eye pupil locations, then cropped and subsampled to 32x32 pixels. The average face is computed and shown in Figure 3.2(a). The resulting orientation map is 31x31 pixels and shown in Figure 3.2(b).

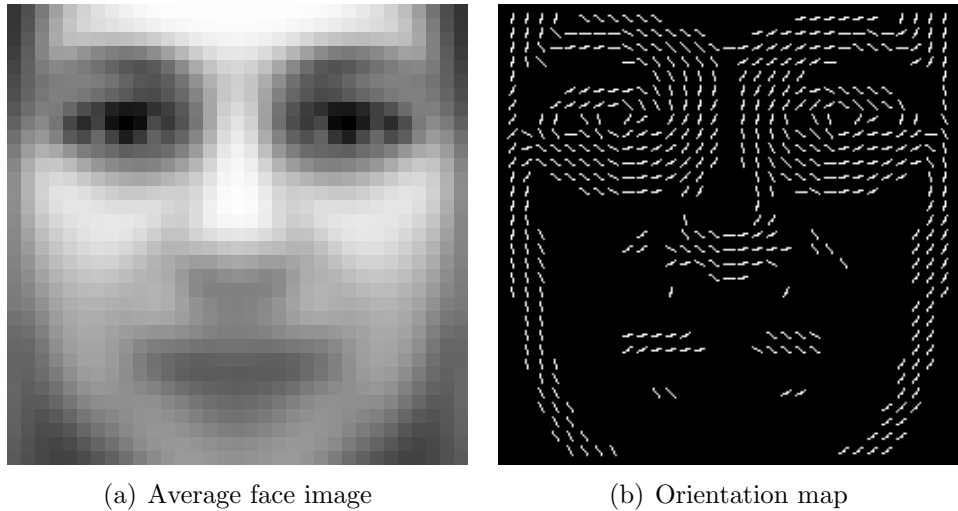


Figure 3.2: Orientation map of the average face

Figure 3.2(b) shows that the main facial features captured by the orientation map

are the outline of the face and the eye regions. Only the orientation vectors with $s(x, y) > s_t$ are shown, because they are the only orientations used by the template when searching an image. The template response is the sum of distance measures $d(v_1, v_2)$ between the template and a candidate image region. Therefore the number of computer operations required to search an image linearly increases with the number of orientation vectors stored in the template, so the total number of vectors should be kept to a minimum. This implementation uses 453 orientations in a 31 by 31 template.

3.3.4 OMD Search

To search an image at multiple scales a Gaussian image pyramid [1] is built. Each level of the pyramid is then searched by computing the match score of the template at all possible sub-windows. The search is speeded up by initially searching at a coarse scale and then performing a refined search if a strong enough match is found. In practice this scheme involves the following algorithm.

Algorithm 3.3 Orientation Map Detector Coarse-to-Fine Search

1. Search every 6th pixel position. Get template response r_1
 2. If $r_1 < T_1$ then test all 8 possible positions that are 3 pixel positions away. Get template response r_2 , for each location.
 3. If $r_2 < T_2$ then test all 8 possible positions that are 1 pixel position away. Get template response r_3 , for each location.
 4. If $r_3 < T_3$ then accept as a face candidate.
-

Note, searching every 6th position instead of every possible position in the image reduces the computation time by a factor of approximately 36. However, this scheme can result in the same candidate being returned more than once. To enable efficient computation appropriate thresholds values T_1, T_2 & T_3 are essential. A large value of T_1 produces a very slow detector! This coarse-to-fine method takes advantage of the

fact that a face template often matches at many locations around and near the face, a phenomenon noted by several researchers e.g. [75][92].

3.4 Normalised Correlation Detector (NCD)

The Normalised Correlation Detector is similar to the Orientation Map Detector. It is a whole face template method, uses a Gaussian pyramid and the same coarse-to-fine search method. However, the method of comparing image subwindows with the face model is different.

3.4.1 Comparing Images using Normalised Correlation

Instead of using orientation maps, normalised correlation is used to compare two images directly. Given two images \mathbf{X} and \mathbf{Y} , with the same dimensions normalised correlation $N_c(\mathbf{X}, \mathbf{Y})$ is defined as follows.

$$N_c(\mathbf{X}, \mathbf{Y}) = \frac{\text{COV}(\mathbf{X}, \mathbf{Y})}{\sqrt{\text{VAR}(\mathbf{X})\text{VAR}(\mathbf{Y})}} \quad (3.11)$$

Similar to the orientation map representation, the normalised correlation measure is insensitive to linear transformations applied to the underlying pixel values of either image.

3.4.2 NCD Training

The NCD is trained on the WEBCAM data set (see Section 3.1). The same average face is computed as when building the OMD (see Figure 3.2(a)). The average face is then rescaled to have zero mean and unit variance, using variance normalisation (see Algorithm 3.4).

Algorithm 3.4 Variance Normalisation of an Image

1. Given a subregion \mathbf{X} of size n , with pixel values $\{x_0, x_1, \dots, x_n\}$
 2. Compute the mean $\mu = \frac{1}{n} \sum_{i=0}^n x_i$
 3. Compute the variance $\sigma^2 = \frac{1}{n} \sum_{i=0}^n (x_i - \mu)^2$
 4. Replace each pixel value $x_i \rightarrow \frac{x_i - \mu}{\sigma}$
-

The normalised average face image is then used to test candidate face regions, as described in Section 3.4.3.

3.4.3 NCD Search

The NCD search uses a Gaussian pyramid and the coarse-to-fine search as described in Section 3.3.4. When testing a subwindow \mathbf{X} of the image the normalised correlation distance measure described in Section 3.4.1 can be simplified due to the face template \mathbf{Y} having zero mean and unit variance. For example $\text{COV}(\mathbf{X}, \mathbf{Y}) = E(\mathbf{X}\mathbf{Y}) - E(\mathbf{X})E(\mathbf{Y})$ and $\text{VAR}(\mathbf{X}) = E(\mathbf{X}^2) - (E(\mathbf{X}))^2$, therefore with $E(\mathbf{Y}) = 0$ and $\text{VAR}(\mathbf{Y}) = 1$, Equation 3.11 simplifies to.

$$N_c(\mathbf{X}, \mathbf{Y}) = \frac{E(\mathbf{X}\mathbf{Y})}{\sqrt{E(\mathbf{X}^2) - (E(\mathbf{X}))^2}} \quad (3.12)$$

In practice, given a subregion \mathbf{X} , $N_c(\mathbf{X}, \mathbf{Y})$ is calculated from the pixel value sums $\sum x_i$, $\sum x_i^2$ & $\sum x_i y_i$ as follows.

$$N_c(\mathbf{X}, \mathbf{Y}) = \frac{\sum x_i y_i}{\sqrt{\frac{\sum x_i^2}{n} - \left(\frac{\sum x_i}{n}\right)^2}} \quad (3.13)$$

Therefore normalised correlation detection is very simple and easy to implement.

This form of detection is very old and is described by Ballard and Brown in their introduction to Computer Vision [1].

3.5 Boosted Cascade Detector (BCD)

The Boosted Cascade Detector [92][93] consists of three parts. The first is an efficient method of encoding the image data known as the “integral image”. The second element is the application of a boosting algorithm known as AdaBoost [26] to select appropriate features that can form a template to model human face variation. The third part is a cascade of templates that allows simple feature sets to quickly discard most of the uninteresting parts of the image.

3.5.1 Integral Images

An integral image is constructed by replacing each image pixel value $i(x, y)$ with a value that corresponds to the pixel sum, above and to the left of the pixel, as shown in Figure 3.3(a).

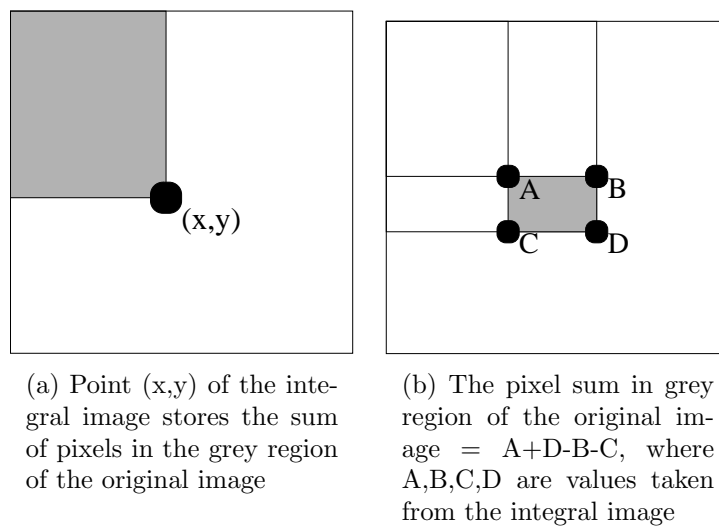


Figure 3.3: The integral image

An integral image can be constructed in one raster scan of the image. This image structure is also known as an “area sum table” and has been used by [18] for computer graphics applications. The structure is useful, because it allows quick calculation of a pixel sum in any rectangular region, independent of the size of the region, as described in Figure 3.3(b).

3.5.2 Feature Classifiers

The ability to calculate area sums efficiently allows the construction of simple efficient classifiers. Each classifier consists of a set of adjacent positive and negative regions. Examples of the different types of feature classifier are shown in Figure 3.4.

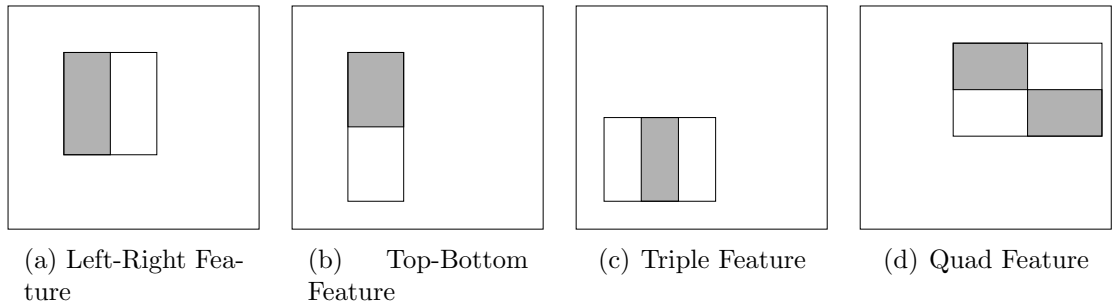


Figure 3.4: Different feature types

Here each feature type is allowed to occupy any sub-rectangle of a face template. In this case the template is 24×24 pixels, which produces approximately 30,000 different feature classifiers. The number of feature classifiers is $O(n^4)$ for a template of size $n \times n$ pixels. Therefore if n is too large, an unfeasible number of potential feature classifiers will be created.

Each individual feature classifier attempts to discriminate between faces/non-faces by summing the difference between adjacent regions. More specifically, a feature classifier $h_j(\mathbf{X})$ consists of a sum of areas response function $f_j(\mathbf{X})$ a threshold θ_j and a parity function $p_j = \{-1, +1\}$ indicating the direction of the inequality sign, as shown in Equation 3.14.

$$h_j(\mathbf{X}) = \begin{cases} 1 & \text{if } p_j f_j(\mathbf{X}) < p_j \theta_j \\ 0 & \text{otherwise} \end{cases} \quad (3.14)$$

Here \mathbf{X} is a 24*24 pixel sub-window of the image, which may or may not contain a face. $h_j(\mathbf{X}) = 1$ indicates that the feature classifier detects a face, $h_j(\mathbf{X}) = 0$ indicates no face is present.

Each feature classifier $h_j(\mathbf{X})$ is very simplistic, so the correct classification rate using individual features will be very low. The next section describes how to combine feature classifiers to make a more effective face model.

3.5.3 BCD - Training with AdaBoost

The Boosted Cascade Detector selects and combines the simple feature classifiers $h_j(\mathbf{X})$ described in Section 3.5.2 to form a face model using an algorithm known as AdaBoost[26].

AdaBoost is an example of classifier boosting [79]. Boosting algorithms aim to combine many “weak classifiers” into one effective “strong classifier”. AdaBoost achieves this by selecting the more promising classifiers. By a process of iteratively re-weighting the training data, it constructs an effective classifier consisting of a large set of appropriately weighted weak classifiers that complement each other and produce better discrimination than any individual weak classifier. AdaBoost is described in Algorithm 3.5.

The training set for each feature classifier $h_j(\mathbf{X})$ is a large set of 24*24 pixel face images (~ 6000 in this implementation) and an equally large set of non-face images. A sample of positive face examples is shown in Appendix C.

The features and weights selected by AdaBoost then form the template that is used to scan the image and detect regions that resemble human faces. In our implementation

Algorithm 3.5 AdaBoost - Adaptive Boosting (based on [92])

1. Given example images $(\mathbf{X}_1, \mathbf{X}_2, \dots, \mathbf{X}_n)$ and labels (y_1, y_2, \dots, y_n) where $y_i = 0, 1$ for negative and positive examples respectively.
2. Initialise weights $w_{1,i} = \frac{1}{2m}, \frac{1}{2l}$ for $y_i = 0, 1$ respectively, where m and l are the number of negatives and positives respectively.
3. For $t = 1, \dots, T$:

- (a) Normalise the weights,

$$w_{t,i} = \frac{w_{t,i}}{\sum_{j=1}^n w_{t,j}} \quad (3.15)$$

so that w_t is a probability distribution.

- (b) For each feature j , train a single feature classifier $h_j(\mathbf{X})$
- (c) Evaluate the error with respect to $w_t, \epsilon_j = \sum_i w_i |h_j(\mathbf{X}) - y_i|$.
- (d) Choose the classifier, h_t , with the lowest error ϵ_t .
- (e) Update the weights:

$$w_{t+1,i} = w_{t,i} \beta_t^{1-e_i} \quad (3.16)$$

where $e_i = 0$ if example x_i is classified correctly, $e_i = 1$ otherwise, and $\beta_t = \frac{\epsilon_t}{1-\epsilon_t}$.

4. The final strong classifier is:

$$h(\mathbf{X}) = \begin{cases} 1 & \text{if } \sum_{t=1}^T \alpha_t h_t(\mathbf{X}) \geq \frac{1}{2} \sum_{t=1}^T \alpha_t \\ 0 & \text{otherwise} \end{cases} \quad (3.17)$$

where $\alpha_t = \log \frac{1}{\beta_t}$

the first 5 features chosen by AdaBoostare depicted in Figure 3.5.

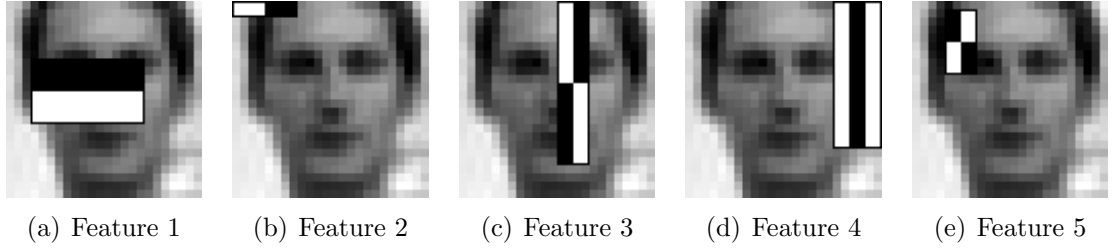


Figure 3.5: Features selected by AdaBoost, overlaid on an example from the training set

The first feature exploits the fact that eyes are generally darker regions than the nose and cheeks. The other features have less obvious interpretation, but generally indicate that the outline of the head is modelled by the feature set. Note the first feature found in our implementation is similar to the first feature selected by Viola and Jones [93], but the second feature is different. This difference is due to variation in the face/non-face training sets used. Also note that the first feature selected by AdaBoost is not symmetric as may be expected from the left-right symmetry of the human face. This is due to asymmetries in the random set of background images which form the non-face training set.

3.5.4 BCD - Search with AdaBoost Model

The face model can now be used to discriminate between image regions containing a human face and background regions, using Equation 3.17 (see Algorithm 3.5). However, the threshold $\frac{1}{2} \sum_{t=1}^T \alpha_t$ is unsuitable, because it is designed to minimise the classification error given an equal distribution of faces/non-faces. When searching images, the vast majority of subwindows do not contain faces, so in practice this threshold must be determined manually using a set of verification face images.

When searching an image the integral image is computed and scanned at multiple scales. The face template is resized at each scale, rather than subsampling the input

image (e.g. using an image pyramid), because using the integral image structure the computational cost of each feature classifier is independent of scale.

Three parameters of the search are the start scale s , the initial search step t and the scale increment i_s , as shown in Figure 3.6.

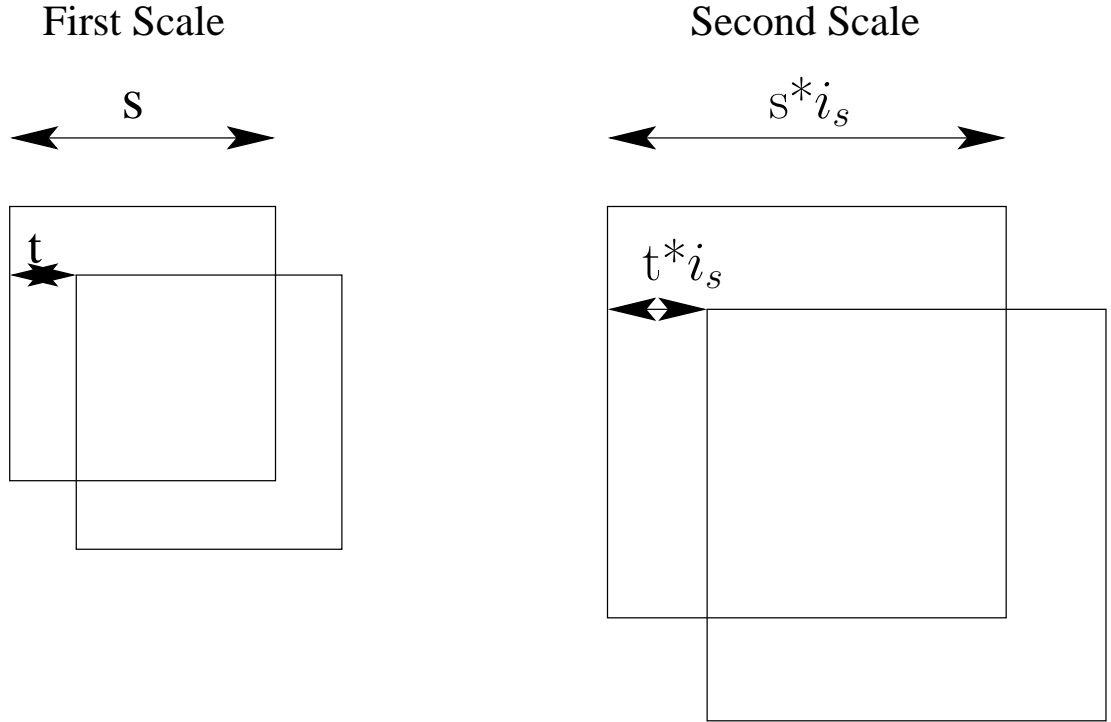


Figure 3.6: Boosted Cascade Detector multi-scale search parameters, start scale s , step size t and scale increment i_s .

The values of s , t and i_s vary with the size of faces that are expected in the image. To find low resolution faces typical values are $s = 24$, $t = 1.5$ and $i_s = 1.25$. The search terminates when the face template is larger than the input image.

To cope with lighting variation, variance normalisation (see Algorithm 3.4) is applied to both the training images (see Appendix C) and each subwindow of the input image. It is easy to variance normalise the positive and negative training sets (see Algorithm 3.4), before applying the AdaBoost algorithm (see Algorithm 3.5). However, during BCD search a more efficient method of variance normalisation is used (see

Algorithm 3.6), which normalises the feature classifier responses $f_j(\mathbf{X})$ (see Equation 3.14) instead of the pixel values directly. This efficient normalisation scheme is one of the reasons for the speed of the Boosted Cascade Detector compared to other template methods e.g. [73] [74] [80], which require the pixels of each subwindow to be normalised before applying the detector.

Algorithm 3.6 Efficient Variance Normalisation

1. Given a subregion \mathbf{X} of size n , with pixel values $\{x_0, x_1, \dots, x_n\}$
 2. Compute integral images for $\sum x_i$ and $\sum x_i^2$
 3. Compute $\sigma^2 = \frac{1}{N} \sum x_i^2 - \left(\frac{\sum x_i}{N}\right)^2$ for subregion \mathbf{X} using both integral images
 4. Compute the sum of areas response $f_j(\mathbf{X})$ for classifier $h_j(\mathbf{X})$ using the $\sum x_i$ integral image
 5. Normalise the response $f_j(\mathbf{X}) \rightarrow \frac{f_j(\mathbf{X})}{\sigma^2}^*$
-

The final speed of the detector is linearly related to the number of features in the template. However the classification performance of the AdaBoost template is also dependent on the number of features. Using more features improves classification performance, but slows down the detector. To use more features, but without jeopardising speed, Viola and Jones developed an algorithm to train a cascade of AdaBoost templates, as described below.

3.5.5 BCD - Training the Cascade

A cascade is employed to reduce image processing time by focusing attention on the more interesting regions of the image. For example the flat regions of an image, clearly do not contain faces and can be quickly discarded by use of a template consisting of only a small number of features. Such a scheme has the potential to greatly improve

*The normalised feature response is independent of the subwindow mean μ , because each feature classifier $h_j(\mathbf{X})$ has an equal number of positive and negative pixels.

the speed of the detector, but still allow a large number of features to be evaluated on highly textured regions that may contain faces. The training of the cascade is described by Algorithm 3.7.

Algorithm 3.7 Boosted Cascade Detector- Building the Cascade

1. Set the level number $i = 0$ and create an empty cascade model C_0 .
 2. If $i = 0$ load from disk a set of face examples P_0 and a set of non-face examples N_0 .
 3. If $i \neq 0$ form a new non-face examples set N_i by running the incomplete cascade C_{i-1} over a set of images, known not to contain human faces.
 4. If the size of set N_i is less than a limit s_t then STOP.
 5. Split the faces/non-faces sets P_0 & N_i into training/verification sets P_{t_0} , P_{v_0} , N_{t_i} & N_{v_i} .
 6. Build template model L_i from face set P_{t_0} and non-face examples N_{t_i} using n_i features[†].
 7. Calculate a threshold t_i by applying L_i to the verification sets P_{v_0} & N_{v_i} , such that the false rejection rate is f_i .
 8. Create C_{i+1} by adding L_i with threshold t_i to the current cascade C_i .
 9. $i = i + 1$
 10. if $i = k$ then STOP else GO TO 2
-

Here the training parameters that need to be set are k , s_t , \mathbf{n} and \mathbf{f} . Where:-

- k is the number of cascade levels
- s_t is the threshold on the minimum size of the non-face training set
- $\mathbf{n} = \{n_0, n_1, \dots, n_k\}$ is the number of features n_i used in each level of the cascade
- $\mathbf{f} = \{f_0, f_1, \dots, f_k\}$ is the false rejection rate f_i at each level of the cascade

[†]This training method is slightly different to the method described in [93], which specifies the minimum acceptable detection rate for each layer rather than the number of features in each layer n_i

In our implementation $k = 16$, $s_t = 1000$, $f_i = 0.01$ for all levels and $\mathbf{n} = \{10, 10, 10, 20, 20, 50, 50, 100, 100, 100, 100, 200, 200, 200, 200\}$. The number of features n_i and the false rejection rate f_i at each level are critical to the speed and the performance of the final cascade. These parameters are found to give good results, comparable to the results presented in [93] on the CMU [75] test data. There is no guarantee that other formulations will not provide better classification performance and be more efficient, e.g. by using a smaller number of features in the first level of the cascade.

However, it is difficult to build multiple models due to the large amount of time required to train the cascade. The main drawback of the Boosted Cascade Detector method is the large amount of training time required to train a model. This extends from the fact that at each stage of the AdaBoost algorithm all possible features must be trained on the entire re-weighted training data. This means on a modern PC (e.g. a 2.0Ghz machine) training the whole cascade requires a few days of computation time. Therefore testing different formulations of the cascade is a time consuming and difficult task.

3.5.6 BCD - Cascaded Search

The BCD cascaded search proceeds in a similar manner to the single template search described in Section 3.5.4. The whole cascade is resized according to the three parameters s, t and f_s described in Figure 3.6. For each subregion a cascade score c_s is computed as follows (see Figure 3.7).

A subwindow is classified as a face if c_s is greater than a fixed threshold. Note returning a score c_s enables a set of candidate regions to be ranked. The original Boosted Cascade Detector gives a binary output of one if the subwindow passes all levels of the cascade and zero otherwise.

The resulting cascaded classifier is extremely efficient. The cascade described in this

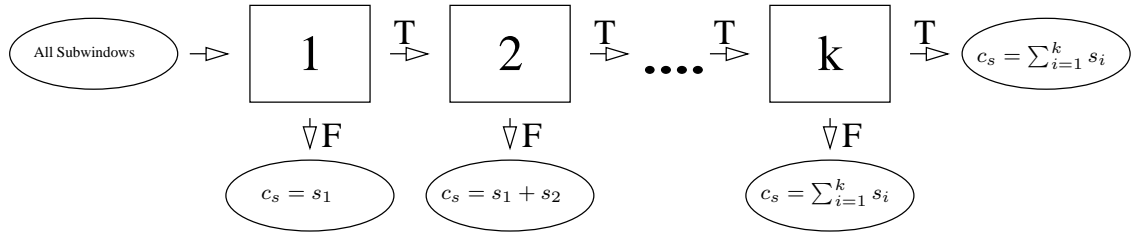


Figure 3.7: Cascaded search, each level returns either true or false. If a subwindow fails a level then no more levels are evaluated. The final score is $c_s = \sum_{i=1}^k s_i$, where s_i is the AdaBoost response[§] from level i .

thesis contains 1470 feature classifiers over 16 levels and is capable of searching a 320*240 image in ~ 400 ms using modest hardware (i.e. a 500Mhz PentiumII processor). This is much faster than searching a test image with a single template containing all 1470 features.

3.6 Face Detection Test Sets

Three publicly available data sets are used to test the four face detection methods. They are the XM2VTS [59], BIOID [43] and CMU [74] data sets.

3.6.1 XM2VTS

A test set is formed from photo sessions 1-4 of the XM2VTS[¶] data set [59]. There are 1817 images (720*576 pixels) containing frontal faces of ~ 200 individuals, taken under controlled conditions against a flat background. The face is large in the image and there is no background clutter, so the face detection task is relatively easy. However the XM2VTS data set does contain many individuals with facial hair and glasses. Some example images are shown below (Figure 3.8).

[§] $s_i = \sum_{t=1}^T \alpha_t h_t(x)$ as defined in Algorithm 3.5

[¶]<http://www.ee.surrey.ac.uk/Research/VSSP/xm2vtsdb/>



Figure 3.8: Example XM2VTS images

3.6.2 BIOID

The BIOID data set was first used by Jesorsky *et al.* [43] and is publicly available^{||}. There are 1521 images of 23 individuals. Each image is 384*286 pixels. The face is large in the image. There is background and lighting variation. Some examples from the BIOID data set are shown in Figure 3.9.

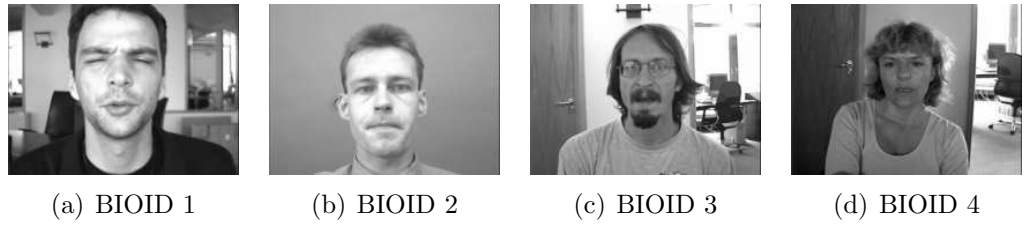


Figure 3.9: Example BIOID images

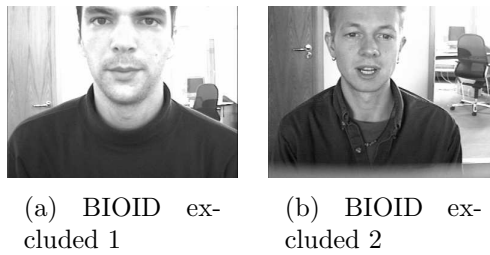


Figure 3.10: Examples of images *not* included in the BIOID subset, as they do not include the whole face region.

This thesis uses two versions of the BIOID data set. The first is the full set of 1521 images. The second version is a subset of 1348 images, which excludes images that do not show the full outline of the face (see Figure 3.10). The BIOID subset is used

^{||}<http://www.humanscan.de/support/downloads/facedb.php>

to test the Boosted Cascade Detector in Chapter 4. Here results are presented on both the full BIOD data set and the subset.

3.6.3 CMU

The CMU data set [74] is a publicly available ** test set that has been used by many face detection researchers [74][73][80] to compare algorithms. The images contain upright frontal faces and are collected from a variety of sources, e.g. scanned photographs from newspapers and the world wide web. The images are generally low quality and contain some very small faces and lots of background clutter and lighting variation. The CMU set is therefore a challenging data set. There are 507 upright faces among 130 images.

The images are divided into 3 groups. Namely sets A,B & C. Sets A+C were collected by Rowley *et al.* [74] and Set B was provided by Sung and Poggio [87]. Set B is also known as the MIT data set and has been used to compare face detection algorithms by itself in the past [65] [88] (see Chapter 2). There is a fourth set of rotated images, which is not used in this thesis. Some examples from the CMU test set are shown in Figure 3.11.

3.7 Face Detection Testing Methodology

The testing methodology varies depending on whether the test set contains one face per image or multiple faces. For single faces, the face localisation accuracy is computed, as described in Section 3.7.1. For multiple faces a FROC curve is computed, as described in Section 3.7.2.

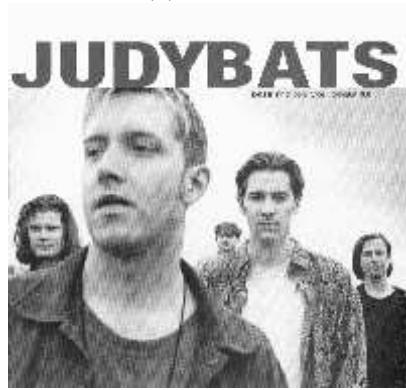
**http://vasc.ri.cmu.edu/idb/html/face/frontal_images/index.html



(a) CMU 1



(b) CMU 2



(c) CMU 3



(d) CMU 4

Figure 3.11: Example CMU images

3.7.1 Face Localisation Accuracy

The XM2VTS and BIODID data sets contain one face per image, so the detection accuracy of a face detector can be measured by searching the image, retaining the highest ranking face candidate and comparing the detected region with the known location of the true face.

For the three template methods (OMD + NCD + BCD) the eye locations are predicted by the average eye locations within the face template. For the Linear Profile Detector the eye locations are the highest ranking candidate eye locations.

A distance metric d_{eyes} , between candidate eye locations and a pair of manually labelled eye locations, is described in Figure 3.12.



Figure 3.12: Distance metric between the eye points predicted by the candidate region and the true eye points $d_{eyes} = \frac{R+L}{2S}$

Hence d_{eyes} is the average error of the right and left eye predictions, divided by the known inter-ocular separation. The cumulative distribution of d_{eyes} over the test set is used to compare different face detectors. For example, see the results of searching the XM2VTS data set in Figure 3.14.

3.7.2 FROC Curve Analysis

For multiple face images (in this case the CMU data set) a Free Receiver Operator Characteristic (FROC) curve is produced. A FROC curve plots the detection rate (i.e. proportion of true faces found) versus the number of false positives detected.

A FROC curve is very similar to an ROC curve, which plots detection rate against false positive rate. However, in this instance an ROC curve is inappropriate because different face detection methods test different numbers of candidate regions when searching the same image. A search algorithm could test many image regions and therefore achieve a very low false positive rate, but still return many actual false positives. Therefore for a given detection rate, only the absolute number of false detections is meaningful when comparing two different methods.

To analyse the CMU data set, each images is searched and a set of candidate regions returned by the face detector. The regions that are close enough to a true face in the image are declared true positives. The value of d_{eyes} for the closest true face is calculated for each detected region, then all candidate regions are classified as either true positives, false positives or ambiguous (not included in FROC curve) according to Table 3.1.

d_{eyes}	Classification
0-0.3	true positive
0.3-0.5	ambiguous
> 0.5	false positive

Table 3.1: Classification of true and false positives. Based on the d_{eyes} metric

Each possible configuration of a face detector can then be used to plot a point on the FROC curve, showing the trade off between the proportion of faces found (i.e. detection rate) and the number of false positives. The FROC curves produced for the Orientation Map Detector and Boosted Cascade Detector are shown in Figure 3.16.

There are some refinements to the list of candidates returned by each detector. For example, candidates that are close together are merged. This reflects the fact that both the Boosted Cascade Detector and Orientation Map Detector produce multiple candidates around a face region, i.e. the face is detected multiple times. This behaviour is also true of some false positive regions. Therefore to produce one candidate per face the candidates in close proximity are merged. The distance metric between any two candidates is described in Figure 3.13.

Two candidates are merged if $d_{cands} < 0.3$. The two candidates are then merged by averaging the top left corners $A\&C$ and the bottom right corners $B\&D$ to form a new candidate. Merging candidates regions ensures that each true face is detected at most once, so is a valid post processing method, producing cleaner search results. It also reduces the number of false positives returned by each detector.

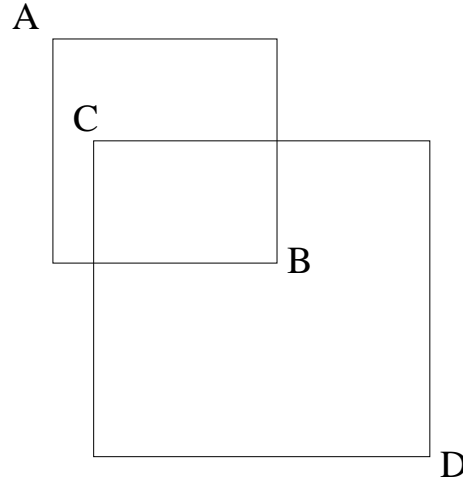


Figure 3.13: The distance metric between a candidate defined by corners (A,B) and a candidate defined by corners (C,D) $d_{cands} = \frac{d_{AC}^2 + d_{BD}^2}{d_{AB}^2 + d_{CD}^2}$, where d_{AB} is the distance between points A & B.

3.8 Face Detection Results

3.8.1 XM2VTS

The cumulative distribution of the eye localisation error (d_{eyes}), when each face detector is applied to the XM2VTS test set is plotted in Figure 3.14.

Figure 3.14 shows that if the proximity threshold is set to $d_{eyes} = 0.3$, then the BCD finds 100%, the OMD 98%, NCD 90% and the LPD 68% of the faces correctly. Therefore at this proximity threshold the BCD is the best performing method, then the OMD followed by the NCD and the LPD. The three template methods (BCD + OMD + NCD) give better results in comparison to the LPD feature based method.

For smaller values of d_{eyes} the relative success of each method is more complex. However, Chapter 6 and Chapter 7 show that the best proximity results are obtained by face detection followed by local feature search. Hence, in Figure 3.14, the success rates for $d_{eyes} < 0.3$ are less important than the success/fail rate at a looser proximity threshold (e.g. $d_{eyes} = 0.3$), because in the later case feature detection accuracy can

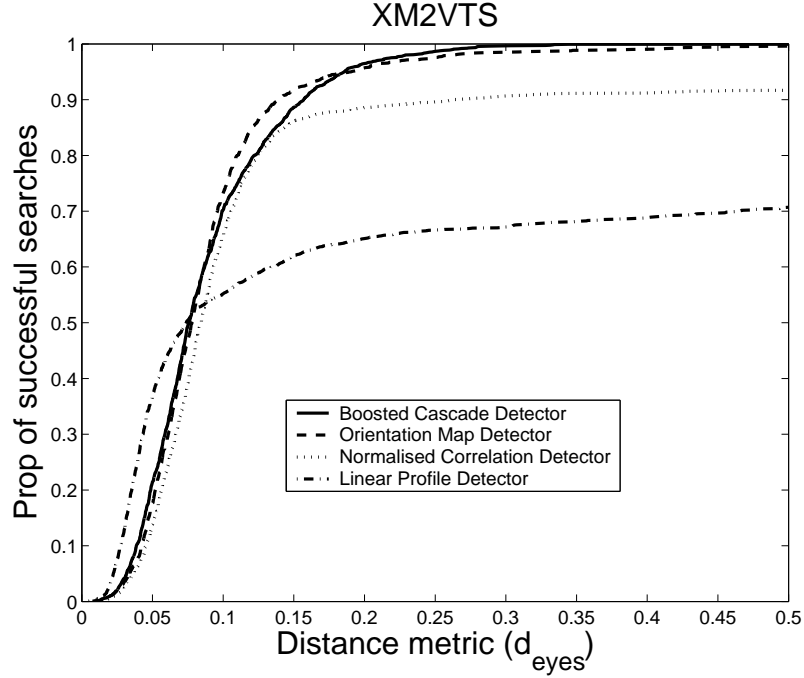


Figure 3.14: Proportion of faces found at different thresholds for d_{eyes} , when searching the XM2VTS test set

be achieved by a more refined local search (see Chapter 6).

3.8.2 BIOID

The success rate with the full BIOID data for each method is shown in Figure 3.15(a). The face detection accuracy on the BIOID test set is generally worse than on XM2VTS test set, for all four methods. This reflects the greater variability of the BIOID data set, especially the effect of more lighting variation. With $d_{eyes} = 0.3$ the BCD gives the best results finding 96% of all faces, the OMD finds 80%, the LPD 61% and the NCD 55%. Therefore the BCD and OMD are still the best performing methods, but with the noisier BIOID data the BCD is significantly better than the OMD. Again, the LPD and NCD perform much worse, but this time the LPD is slightly better than the NCD.

When tested on the BIOID subset (Figure 3.15(b)), which excludes images that do

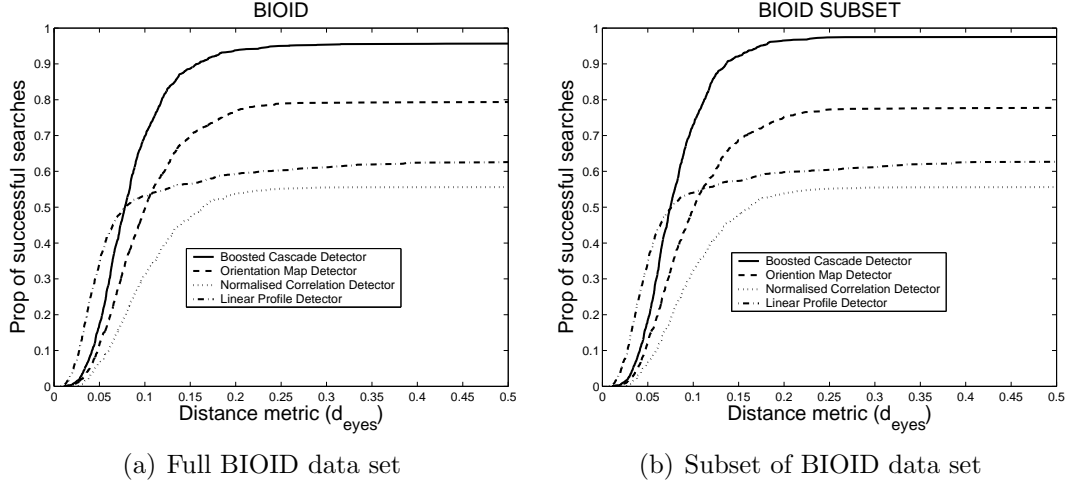


Figure 3.15: Proportion of faces found at different thresholds for d_{eyes} , when searching the BIOID test set

not show the full face outline, the performance of the BCD is slightly improved $96\% \rightarrow 98\%$ ($d_{eyes} = 0.3$). This is because the relatively large face region modelled by the BCD is not able to locate faces overlapping the edge of the images (see Figure 3.10). The performances of the OMD, NCD and LPD on the full BIOID and BIOID subset are very similar, because these detectors model a smaller face region. The two graphs in Figure 3.15 show that the BCD gives by far the best better performance overall on the BIOID data set.

Note that in Section 7.7 of this thesis, the edge effects on the full BIOID data set are alleviated by replicating the edge pixels to form a border around each BIOID image. Thus it may be possible to further improve the performance of the BCD in Figure 3.15(a), relative to the other methods by using this approach.

3.8.3 CMU

The CMU data set contains multiple faces per image, therefore a simple face localisation analysis cannot be performed. Instead a FROC curve is computed for both the two best performing detectors (i.e. the BCD+OMD), see Figure 3.16.

The FROC curve shows that the BCD is capable of finding 88% of faces in the

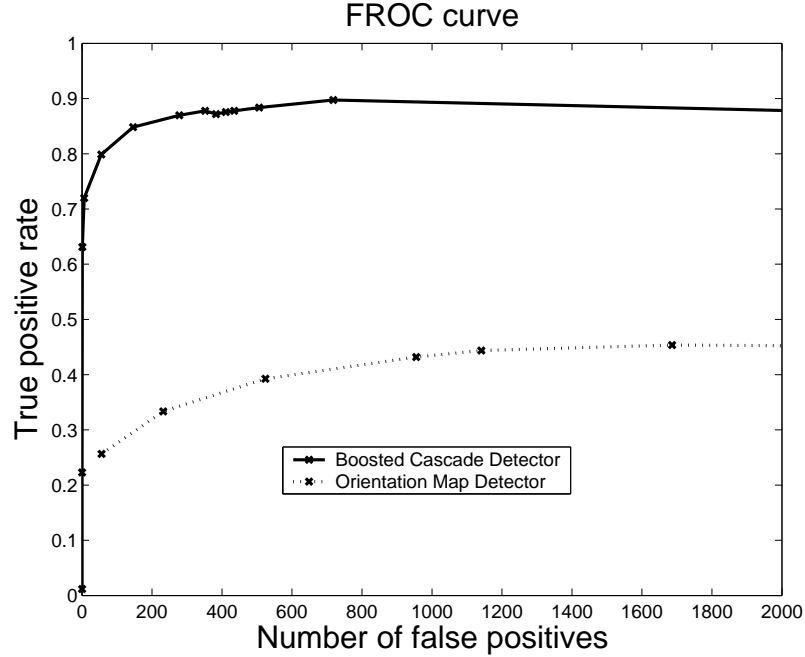


Figure 3.16: FROC curve for the Boosted Cascade Detector and Orientation Map Detector when applied to the CMU test set

CMU data set, with 350 false detections (i.e approx 2.7 false detects per image). Considering the challenging nature of the CMU data set, this is a good result^{††}. In contrast the OMD is only able to detect $\sim 35\%$ of faces for 350 false detections. Therefore the OMD is unable to cope with the variability of the CMU test set.

Note that in Figure 3.16 the FROC curve for the BCD first increases and then decreases. This is due to the merging of similar face candidates, described in Figure 3.13. As the number of false positives increases more face candidates are merged, which reduces the number of faces found in some images, which can result in a lower true positive rate. Example search results, using the Boosted Cascade Detector and merging the similar candidates, are shown in Figure 3.17.

The BCD is also shown to give superior performance to the OMD on the FGNET “Smart meeting”^{‡‡} data set [14], where a similar FROC curve analysis is performed.

^{††}Viola and Jones quote a detection rate of 93% with 350 false detections [93]. This superior result is probably due to differences in the training set and more finely tuned cascade parameters.

^{‡‡}<http://www.cvg.cs.rdg.ac.uk/PETS-ICVS/pets-icvs-db.html>

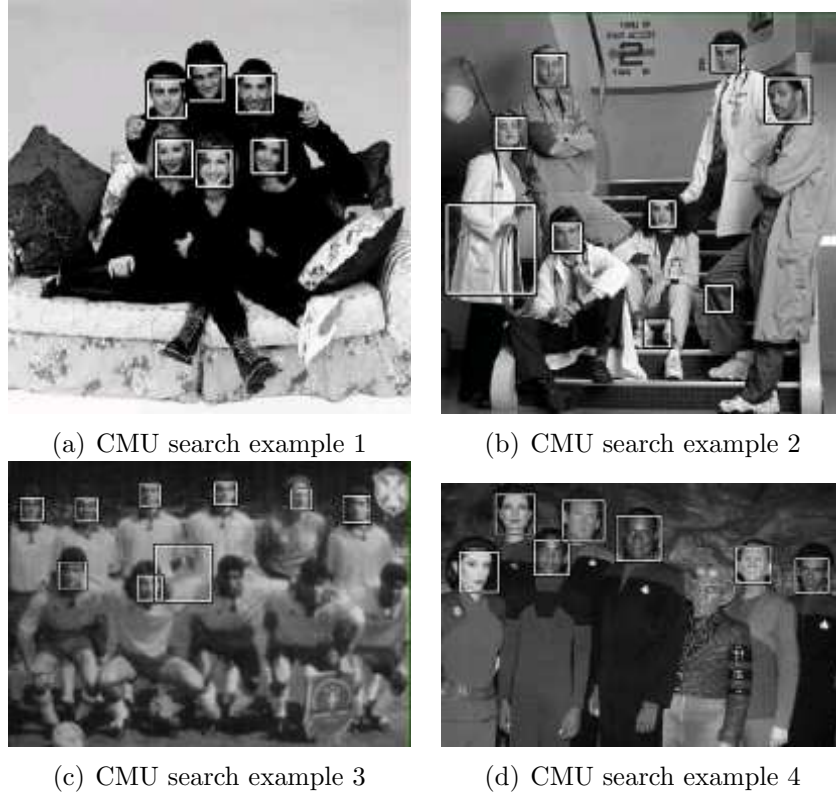


Figure 3.17: Example search results on the CMU images

However, these results only use the basic OMD formulation [31]. In later work Fröba [32] demonstrated more successful results on the CMU data set, by combining the OMD with the SNoW based detector due to Roth *et al.* [73]. More recently, Fröba [28] applied AdaBoost to select discriminating orientation vectors, which also improves on the original approach (see literature review in Section 2.2 for more details). However, the basic OMD detector, as presented here, performs poorly on the CMU data.

3.9 Timings

The time to search an individual image using each method, when applied to the BIOID data set is shown in Table 3.2.

The LPD is by far the fastest method, but also the least reliable. The OMD is

Method	Search Time
Linear Profile Detector	30ms
Normalised Correlation Detector	200ms
Orientation Map Detector	250ms
Boosted Cascade Detector	400ms

Table 3.2: Time to search BIOID image (384*286 pixels) using a 500Mhz PII processor

marginally faster than the BCD, but the BCD gives much superior performance. The speed of the NCD + OMD relative to the BCD is due to the coarse-to-fine search method (see Algorithm 3.3). However, this approach could probably be adapted for the BCD to make it even more efficient.

3.10 Conclusions

The BCD gives far superior performance compared with the OMD, NCD and LPD. On the relatively easy XM2VTS test set, the BCD give similar results to the OMD. On the difficult BIOID+CMU data sets, the BCD is shown to be much more successful than the OMD.

The main advantage of the BCD over the OMD is an increased ability to detect slightly rotated faces. This is due to the non-linear nature of the AdaBoost templates used in the Viola-Jones cascade. This compares to the linearity of the Orientation Map Detector template, which only models exactly frontal views of the face.

The LPD is shown to be the quickest method, but also the least accurate. The BCD+OMD+NCD are slower, but still efficient methods. The reliability and speed of the BCD means that this method is utilised and investigated in more detail in the rest of this thesis.

Chapter 4

Boosted Cascade Detector Experiments

This chapter describes experiments with the Boosted Cascade Detector (BCD). Comparisons are made between cascaded AdaBoost models and single AdaBoost models (see Section 4.2). The number of features in each model, size of training set and face region modelled are varied (see Sections 4.3, 4.4 and 4.6). In Section 4.7 models are built to detect individual facial features.

4.1 Benchmark Detector

To show the effect of different build parameters on the BCD it is necessary to build many models with the same training set (WEBCAM, see Section 3.1) and evaluate them on the same test set (BIOID subset, see Section 3.6.2). An initial model is built using the following set of parameters.

- 10 features per level (10 levels in the cascade)
- 1055 images in the training set

- 21*21 pixel template resolution
- 10 samples of each original face image

In the rest of this chapter these parameters are varied from the initial model and compared with the performance of the benchmark detector.

Note the WEBCAM data set contains only 1055 original face images compared to ~ 6000 for the face detector described in Chapter 3. However, it is possible to over-sample each image to artificially generate more training examples. The effect of over-sampling a face image is shown in Figure 4.1.

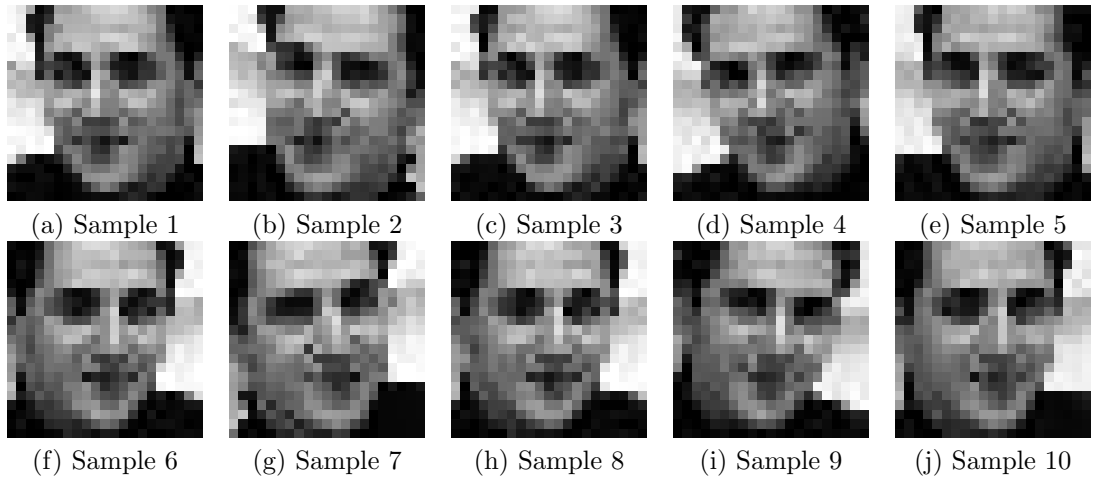


Figure 4.1: Reflecting and over-sampling a face image

The original image is sampled to the appropriate size (21*21 pixels). Then the image is resampled 4 times at a random orientation in the range $\pm 20^\circ$ and at a random scale in the range $\pm 10\%$. The reflected image is sampled in the same manner to generate 10 training examples from the original one. The effectiveness of this over-sampling technique is evaluated in Section 4.4.

When training the initial model on the WEBCAM data, the first 5 features selected by AdaBoost are shown in Figure 4.2.

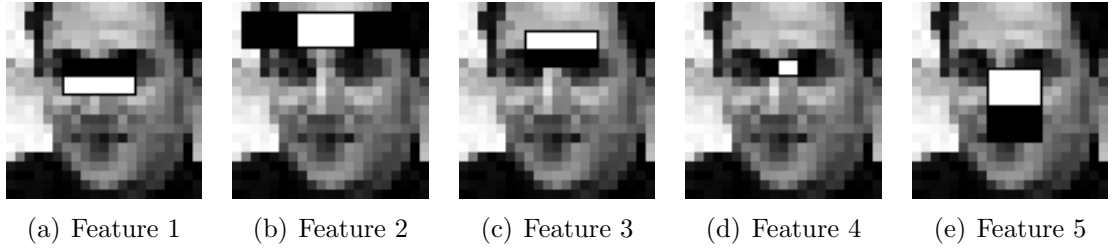


Figure 4.2: Features selected by AdaBoost, overlaid on an example from the training set

The first feature selected is similar to the first feature selected for the Boosted Cascade Detector in Chapter 3, see Figure 3.5. The darkness of the eye region relative to the cheek region below is again emphasised as a key facial characteristic. However the features are generally different due to varying training sets and sampling the face to 21×21 pixels instead of 24×24 pixels.

The other major difference between benchmark detector and the Boosted Cascade Detector created in Chapter 3 is that the latter uses vastly more features - 1470 split over a 16 level cascade. In contrast, the benchmark detector has only 100 features split over a 10 level cascade. Therefore the classification performance of this new “slimmed down” cascade is expected to be worse. The performance drop is shown in Figure 4.3.

Figure 4.3 shows that the original 1470 feature cascade (24×24 pixels) is more reliable than the new 100 feature cascade (21×21 pixels). At a proximity threshold of $d_{eyes} = 0.3$ the success rate on the BIOD data set is 90% for the benchmark detector compared to 98% with the old BCD from Chapter 3. Therefore the benchmark detector used in this chapter is not an especially good face detector. However, this initial model provides a baseline to compare the effect of varying the training parameters, such as the number of features and training set size.

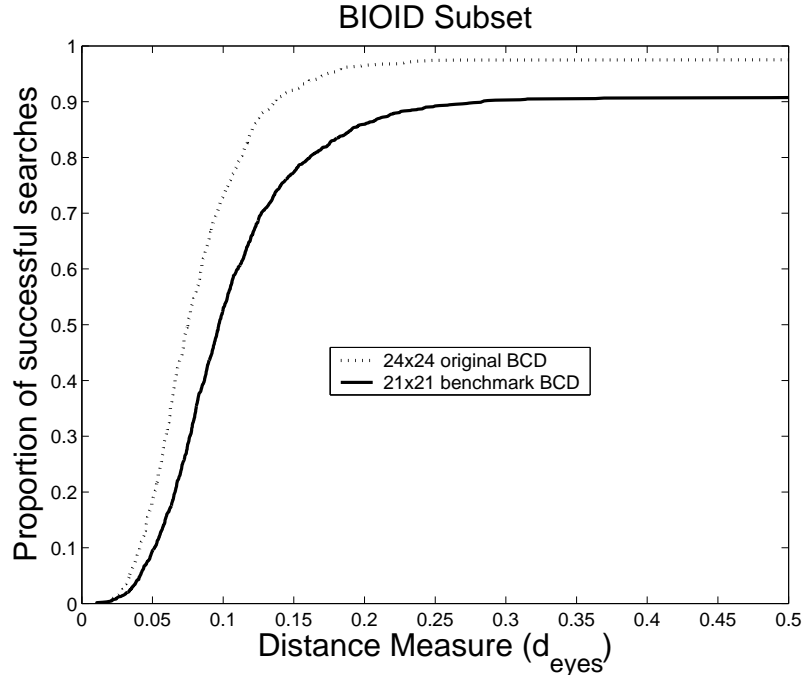


Figure 4.3: Proportion of successful searches at different thresholds for d_{eyes} , when searching with the (16-level 24x24 pixel) Boosted Cascade Detector from Chapter 3 and the new (10-level 21x21 pixel) benchmark detector

4.2 Comparison with Single AdaBoost Model

The benchmark detector with 100 features spread over 10 levels is compared with a single AdaBoost model built with 100 features in a single level. The performance of the cascade versus the single model is shown in Figure 4.4.

Figure 4.4 shows virtually no difference in the location accuracy of the 10 level cascade and the single 100 feature AdaBoost model. However, the cascade is much faster than the single model. This is because candidate regions can be rejected after just 10 feature evaluations, compared to 100 feature evaluations for each candidate region with the single model. In practice the 10 level cascade searches an image from the BIOID data set in $\sim 150\text{ms}$ compared to $\sim 500\text{ms}$ with the single model. Therefore the computational speed is increased by 70% using the cascade, with no loss in performance. This result is similar to that reported by Viola and Jones [93].

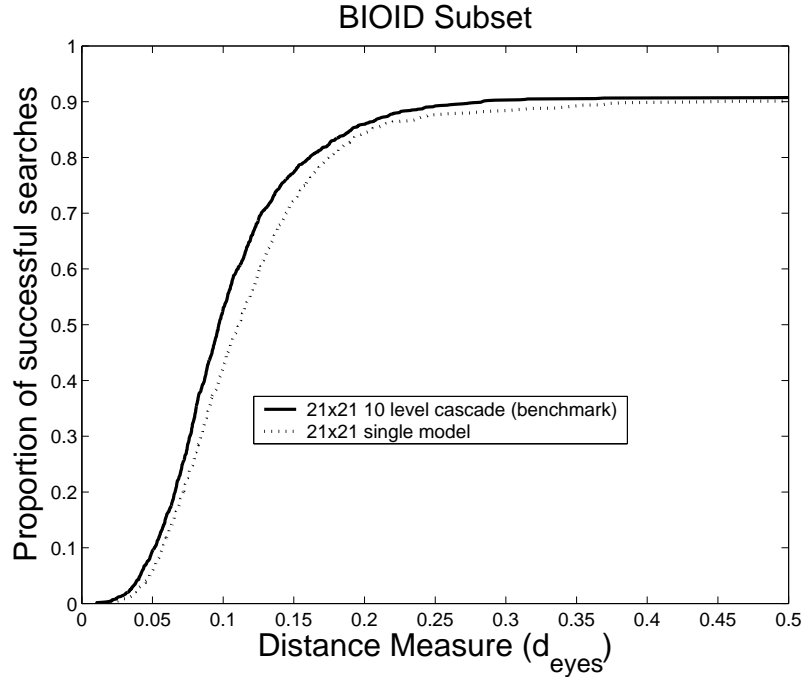


Figure 4.4: Proportion of successful searches at different thresholds for d_{eyes} , when searching with the single template and cascade

4.3 Vary Number of Features

The number of features in each level of the cascade can be varied. Figure 4.5 shows the performance of cascades using 3, 6, 10, 20 and 50 features in each of the 10 levels.

The graph shows that the greater the number of features in each level, the greater the detection accuracy. For example with $d_{eyes} = 0.3$, only 63% of faces are found using 3 features per level, 90% with 10 features, increasing to 97% of faces using 50 features.

There is a trade off between accuracy of detection and speed. The 10Lx10F benchmark detector searches a BIOID image in $\sim 150ms$ compared to $\sim 400ms$ with the 10Lx50F cascade. However, the 10Lx3F cascade searches an image in $\sim 220ms$, i.e. slower than the 10Lx10F cascade.

Therefore the number of features in each level is critical to the overall performance, but obtaining a suitable trade of between classification and speed is only possible by

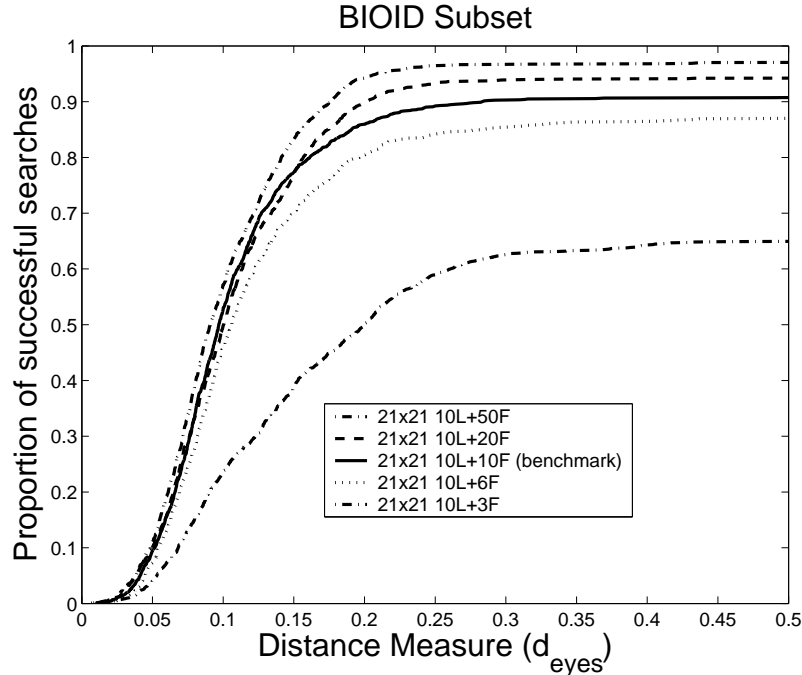


Figure 4.5: Effect of varying the number of features in each level

trial and error. The optimal configuration may also be dependent on the size and nature of the training data. Generally using more features improves classification performance for each level, but slows down the detector.

4.4 Vary Training Set Size

The effect of training set size was tested by building the 10Lx10F cascade using the full set of 1055 WEBCAM faces and comparing the performance using a subset of 500, 200, 50 or just 10 faces, see Figure 4.6(a). Here each training image is still over-sampled 10 times, as described in Section 4.1.

Figure 4.6(a) shows that with $d_{eyes} = 0.3$ using 1055 faces finds 90% of faces, using 500 faces finds 84% and 200 faces finds 80%. Therefore the more original face images in the training set the better. The detector performance drops dramatically when using as few as 50 or even just 10 faces in the training set.

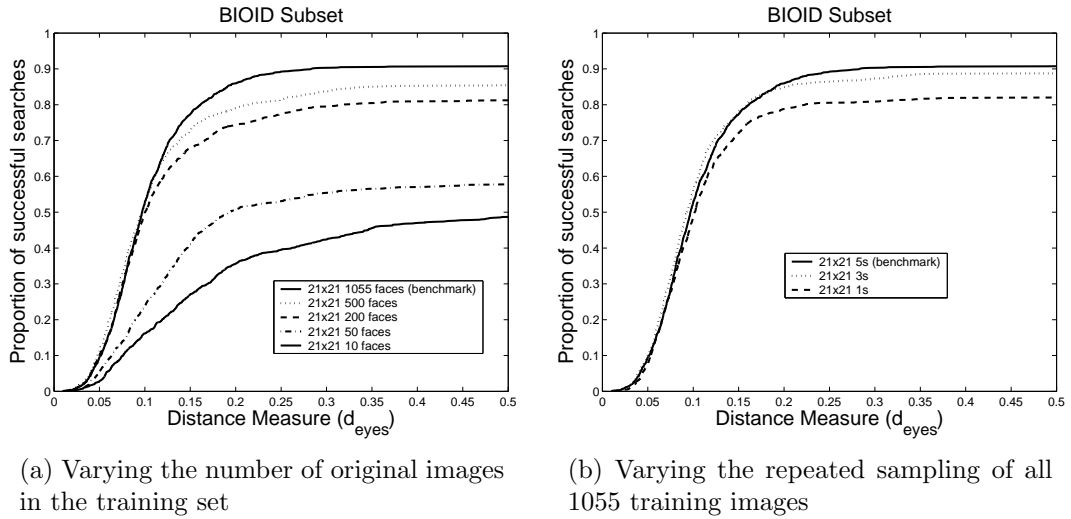


Figure 4.6: Varying the training set size and resampling

The size of the training set can also be changed by varying the number of times each individual training face and its reflection is resampled. Figure 4.6(b) shows the effect of reflecting and over sampling all 1055 WEBCAM training images once, 3 times and 5 times (the initial setting).

With $d_{eyes} = 0.3$ the original cascade, sampling 5 times, finds 90% of faces. When sampling 3 times, this reduces to 88% and when resampling is not used only 80% of faces are found. Therefore by sampling the same image multiple times we are able to artificially expand the data set and provide better performance.

4.5 Vary Template Resolution

The template resolution of the face region can also be varied and used to build different face detectors. For example, Figure 4.7 shows the same training set image region sampled to different sizes.

The BCD can therefore be built using a 21x21, 18x18, 15x15 or 12x12 pixel template. The sampling rate changes the training set images, but also changes the number of possible feature classifiers that AdaBoost may choose from to create the strong

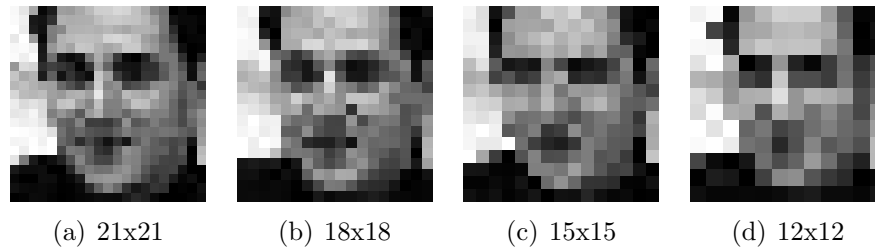


Figure 4.7: Varying face template resolution

classifier. The number of possible features is $O(n^4)$ for a template of size $n * n$ pixels. In the author's implementation the number of possible feature for each template is shown in Table 4.1. The performance of each of the four models is shown in Figure 4.8.

Template	21x21	18x18	15x15	12x12
No. Features	16,800	8,829	4,048	1,593

Table 4.1: Number of possible features during training, given different size templates

Figure 4.8 shows that using an 18x18 template produces similar result to the original 21x21 cascade, 89% compared to 90% (using $d_{eyes} = 0.3$). When using a 12x12 template the detection accuracy drops to 85%. However when sampling to 15x15 pixels the detection rate actually increases to 94%.

This shows that there is no simple relationship between template resolution and classification performance. The 15x15 model may perform better because it learns a lower resolution model that is more stable between different individuals. It may be that with a relatively small training set (i.e. 1055 faces) the 21x21 model over learns spurious small details that do not generalise across individuals. The 12x12 template may have too low a resolution to learn the characteristics of the human face, or too few potential feature classifiers (only 1,593 versus 16,800 using a 21x21 template) for the AdaBoost algorithm to work effectively.

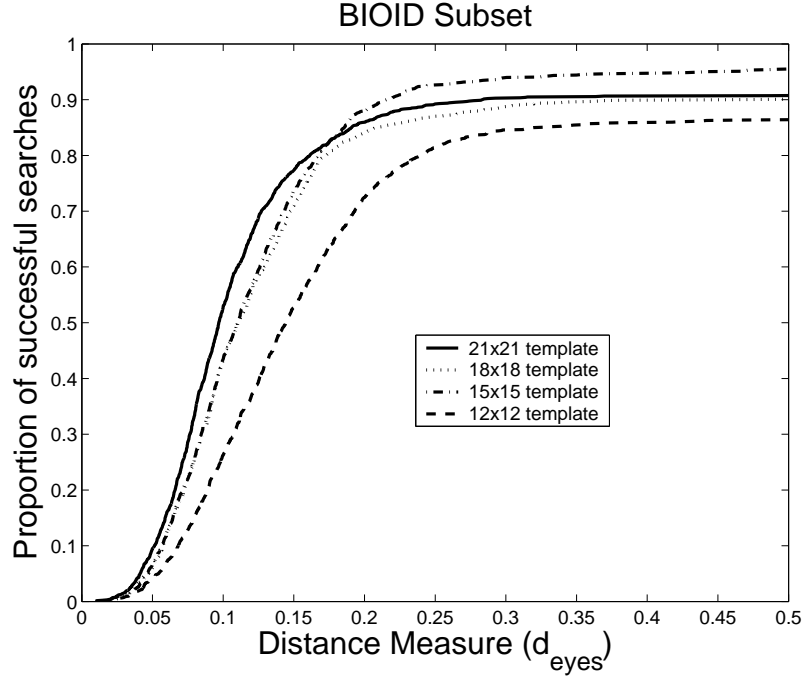


Figure 4.8: Varying the template resolution

4.6 Vary Face Region

The detectors in Section 4.5 subsample the training data to different size templates. However the region modelled is constant (see Figure 4.7). Instead of modelling the whole face and outline, it is also possible to subsample cropped face images and merely attempt to model the internal face features. Figure 4.9 shows a cropped training image subsampled to various template resolutions. The performance of each cascade using cropped regions, versus the original whole head region is shown in Figure 4.10.

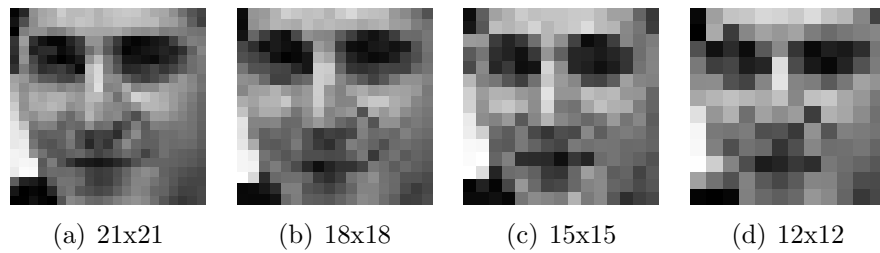


Figure 4.9: Cropped face regions at various resolutions

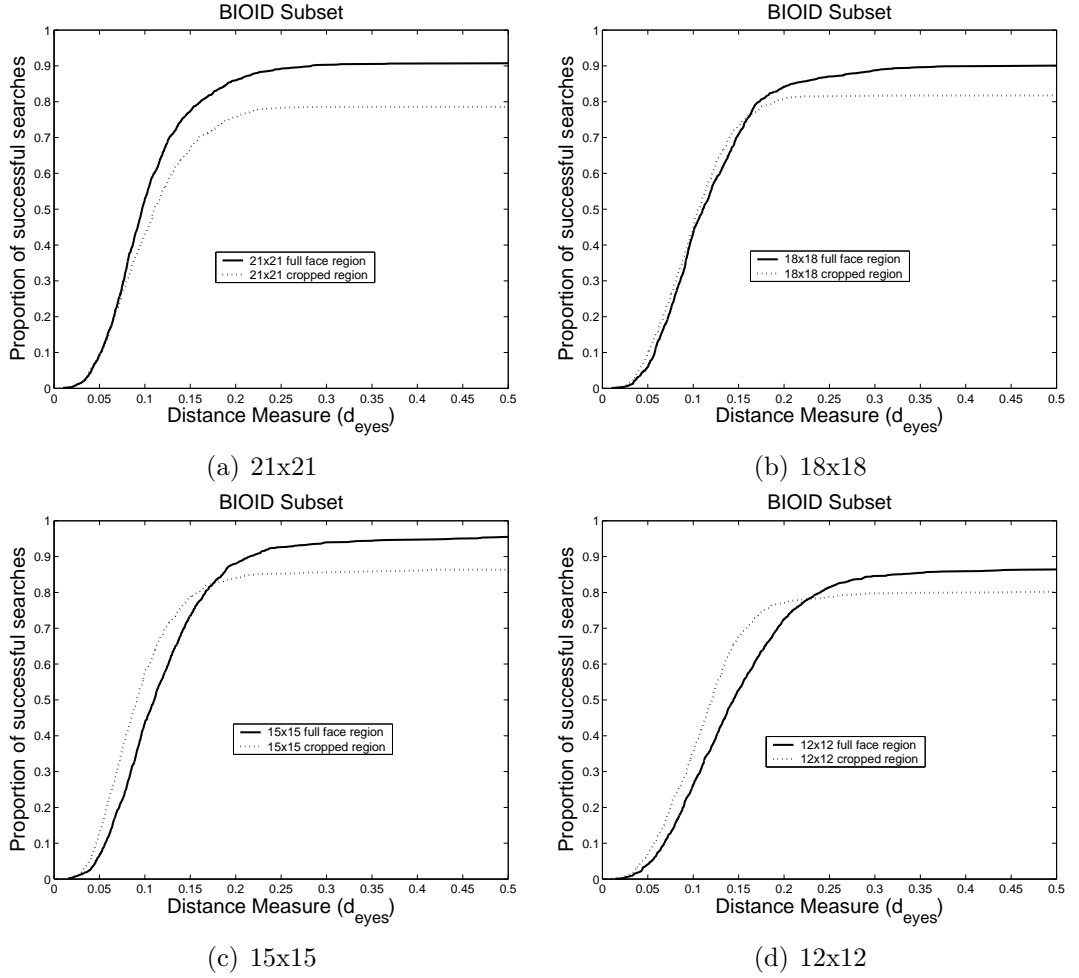


Figure 4.10: Cropping the face template region

The four graphs of Figure 4.10 show that the original whole head template outperforms the cropped face template at all pixel resolutions, by 5-10%, using $d_{eyes} = 0.3$. Therefore it is definitely worthwhile modelling the face outline as well as the internal face features. This result agrees with Viola and Jones findings that the whole head region provides more robust models [93].

4.7 Single Feature Detectors

The face template region can be altered more dramatically. For example by searching for individual facial features, instead of the whole face. This technique is illustrated

by building cascaded models of the right eye region. The template regions are shown in Figure 4.11.

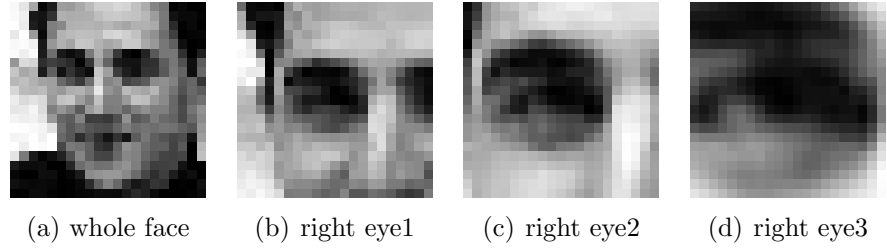


Figure 4.11: Varying right eye template region

Figure 4.11 shows the original whole face region. The other three regions are centred on the right eye, but covering a progressively smaller proportion of the face. All templates are subsampled to a resolution of 21x21 pixels. The performance of each of the four cascades is shown in Figure 4.12

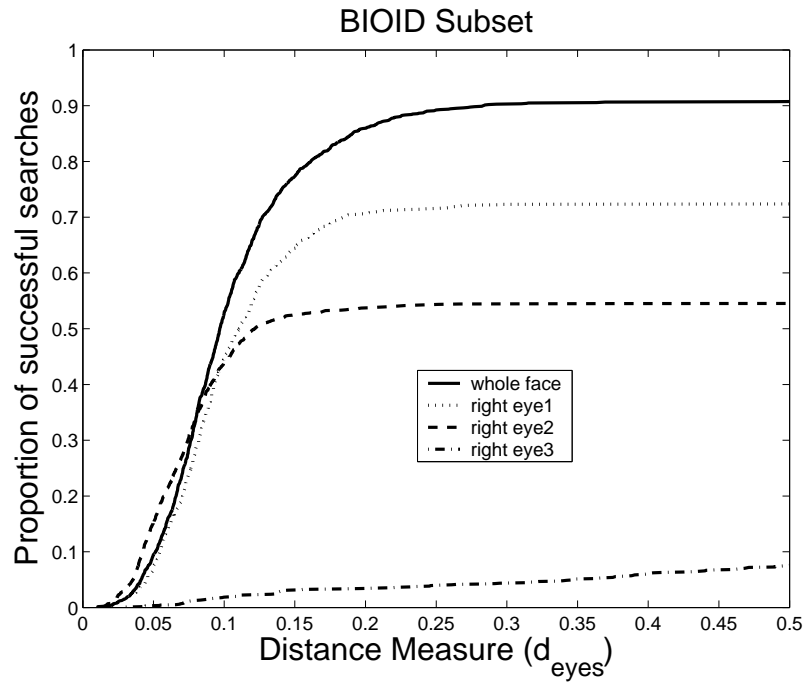


Figure 4.12: Comparing whole face detection with various right eye detectors

Figure 4.12 shows that the whole face template is a far more robust than any of the

right eye templates. With $d_{eyes} = 0.3$, the whole face detector (see Figure 4.11(a)) find 90% of faces. When using the first right eye region (shown in Figure 4.11(b)) the detection rate drops to 72%. As the template region is shrunk to include just the eye (see Figure 4.11(c)), the face detection performance decreases. With the smallest right eye region (see Figure 4.11(d)) only 5% of faces can be found (with $d_{eyes} = 0.3$). Therefore individual feature detection is considerably less reliable than whole face detection, using the BCD. Methods for constraining the feature search and therefore improving the detection of individual features are discussed in Chapter 6 of this thesis.

4.8 Extensions to the BCD Literature Review

There have been various extensions to the Boosted Cascade Detector method published since the original paper by Viola and Jones [92]. The results presented in this thesis use the original formulation, but it may be possible to improve results by adopting alternative methods, as discussed below.

For example, Viola and Jones later described a variant of AdaBoost, which they call “Asymmetric AdaBoost” [91]. This algorithm selects and weights weak classifiers using a cost function which penalises false negatives more than false positives (hence asymmetric). The original AdaBoost algorithm [26] performs feature selection, weighting false positives and false negatives equally, which is inappropriate when forming a cascade of classifiers. Viola and Jones demonstrate marginally improved results using asymmetric boosting compared to symmetric boosting [91].

A number of alternative boosting schemes and an expanded set of features are investigated by Lienhart *et al.* [52]. Eleven new feature types are proposed in addition to the original four (see Figure 4.13). The expanded feature set is shown to give more efficient cascades when modelling faces, i.e. fewer classifiers in each level and is capable of improved classification performance on objects which have more diagonal

structure [53].





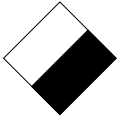
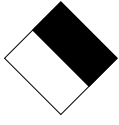

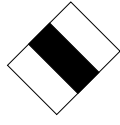
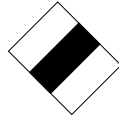


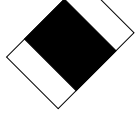
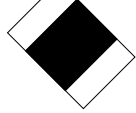

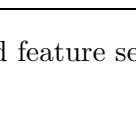

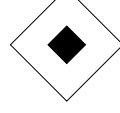
orig					
extra					
					
					

Figure 4.13: Expanded feature set used by Lienhart *et al.* [53]

Lienhart *et al.* [53] also compare three different variants of AdaBoost, namely Discrete AdaBoost, RealBoost and Gentle AdaBoost. Discrete AdaBoost is the original approach described by Schapire and Freund [26]. RealBoost is a variant which considers the margin of each weak classifier [84]. Gentle AdaBoost fits a regression function using least squares to create individual classifiers [27]. With their formulation Lienhard *et al.* find that Gentle AdaBoost is superior to other forms of boosting when building the cascade, both in terms of classification performance and speed. Lienhart *et al.* also experiment with CART tree classifiers [3] compared to individual feature classifiers and demonstrate a small improvement in performance.

Li *et al.* [49] generalise the feature set to include non-adjacent regions (see Figure 4.14). Li *et al.* use RealBoost [84] to perform feature selection and employ a backtracking scheme known as FloatBoost [69], to allow features to be deselected at a later stage, if they are shown to be redundant relative to other features. They demonstrate marginally better results using FloatBoost compared to conventional RealBoost. Li *et al.* also describe a multi-view face detection method, for detecting profile faces, see Section 2.3 for a description of this approach.

More recently Wu *et al.* [97] use RealBoost and Haar wavelet features, but describe a

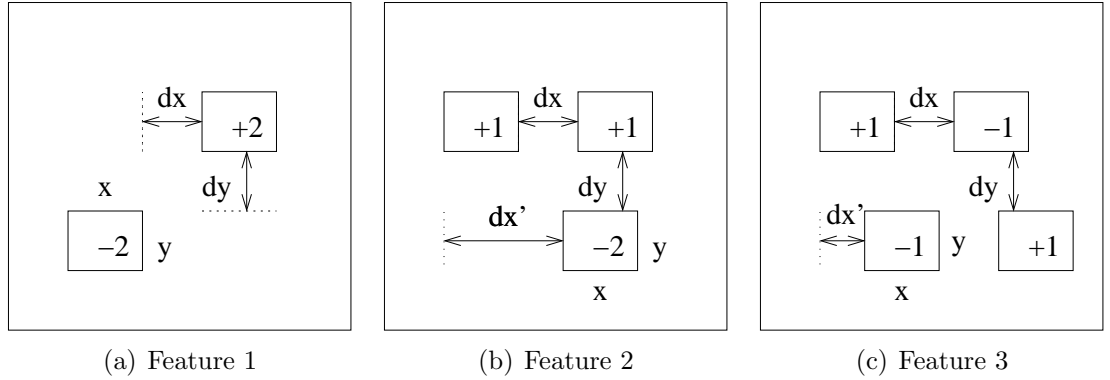


Figure 4.14: Feature types used by Li *et al.*[49]

“nesting-structured cascade”. This approach is the same as the original Viola-Jones cascade formulation, except each classification layer is retained as the first weak classifier in the next layer of the cascade. This method retains the discriminating power of the early classification layers, rather than throwing the information away leading to smaller more efficient cascades and better classification performance. Wu *et al.* present improved results compared to the original Viola-Jones results on the CMU upright test set. Wu *et al.* also extend the Viola-Jones frontal face detector to build a true multi-view face detector, i.e. a detector capable of detecting faces which are simultaneously rotated in-plane and out-of plane (see Section 2.3 for more details).

The face region adopted by most researchers generally includes the face outline in the template. This is found to be more robust than merely modelling the internal face features [93]. The size of the template region is generally selected as 20x20 pixels. The original Viola paper [92] used a 24x24 pixel region. However, several authors claim 20x20 pixels to be optimal [49] [52].

Finally Kruppa *et al.* [47] demonstrate that the template region used by the Boosted Cascade Detector can be expanded to include the head and shoulders, as well as the face region. Kruppa demonstrates that in low resolution images, where face detection is not possible this approach is able to improve on the face only detection approach, mainly due to the distinctive shape of the human neck and shoulders.

4.9 Conclusions

This chapter demonstrates the effect of various build parameters on the BCD. The main problem with the BCD is the vast number of configurations which are possible and the large amount of time required to train the detector. Therefore in practice it is impossible to create the optimal detector. However the experiments performed on the WEBCAM training set and BIOD test images have shown various important aspects of the training method. As follows:-

Cascade vs Single AdaBoost classifier

Using 100 features spread over a 10 level cascade produces similar detection performance to using one 100 feature classifier. However, the cascaded classifier is approximately 4x quicker than the single classifier. A similar result was found by Viola and Jones [93].

Vary Number of Features

The number of features used in each level of the cascade improves the classification accuracy of each level and therefore gives a significant improvement to detection accuracy. However, increasing the number of features in each level also slows down the detector.

Vary Training Set Size and Sampling

The larger the training set the better. Sampling the training set to generate artificial examples improves performance with 1055 examples, but the number of original images is the most critical factor.

Vary Template Resolution

The resolution of the AdaBoost template does affect performance. However larger resolution templates are not necessarily better than lower resolutions. The best resolution may be dependent on the training set.

Cropping the Face Region

The original template region models the whole face including the outline. Modelling a cropped region just including the face gives worse results. This result was also found by Viola and Jones [93].

Single Feature Detectors

Cropping the template region even further, to individual facial features reduces the detection performance even more. A detector modelling the right eye only, gives extremely poor performance. This indicates that local feature detection is likely to be non-trivial, using the BCD.

The last section of the chapter outlines recent advances relating to the BCD. The most comprehensive experiments are performed by Lienhart *et al.* [52]. They demonstrate that using Gentle Boosting may give a significantly better result compared to Discrete Boosting. However in most circumstances, e.g. the face viewed using a web camera mounted on a computer, with reasonable lighting, the original approach due to Viola and Jones is shown to be more than adequate. Therefore the plain vanilla version of the BCD is used for face detection in this thesis.

Chapter 5

Shape Modelling

The right eye region detector investigated in Chapter 4 is shown to be unreliable. However, if many local feature detectors are applied to a constrained face region it may be possible to reduce the number of false detections by only selecting plausible configurations of feature matches. This chapter describes shape modelling techniques and methods of assessing the likelihood of a candidate set of facial feature locations.

5.1 Suitable Landmarks

To model face shape, independent of face texture, a set of facial features must be selected and manually labelled across a set of face images. All twenty landmarks from the 1055 WEBCAM images were used (see Section 3.1) to build shape models. These twenty features were selected because they are easily identifiable by a human operator and present in all human faces. For example the corners of the eyes and mouth are unambiguous and salient features. These landmarks then form correspondences consistent between individuals (see Figure 5.1).

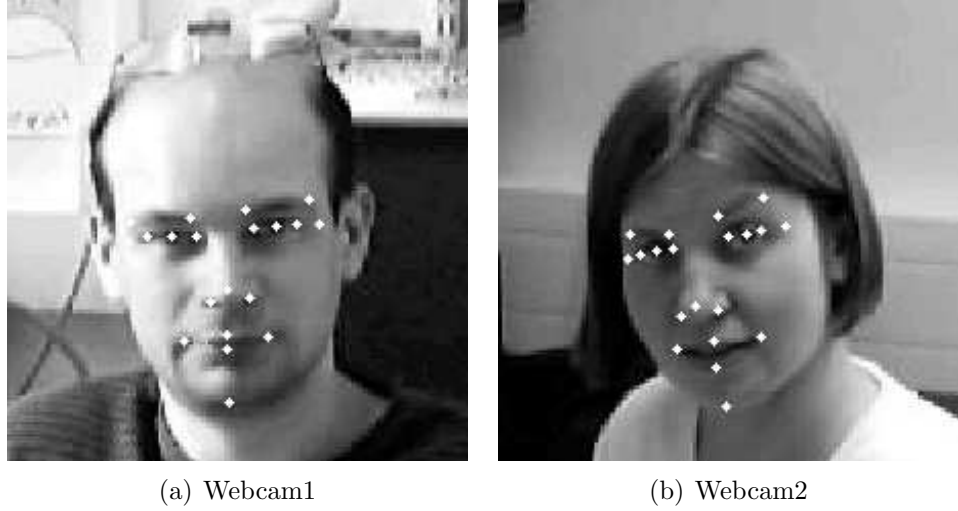


Figure 5.1: Corresponding landmark points between 2 examples from the WEBCAM data set

5.2 Intrinsic Shape Variation

Shape is defined as “The geometrical information that remains when location, scale and rotational effects are filtered out from an object” [20]. Under this scheme a shape is considered unaltered if it is scaled, rotated or translated. Two shapes are only considered identical if they can be mapped to each other using these three transformations. For example, any pair of equilateral triangles are considered the same, however, an isosceles triangle (non-equilateral) is considered to have different shape.

The intrinsic shape variation of an object is the variability that is not due to scale, rotation and translation. The object, in this case the human face, is represented by a set of example shapes extracted from manually labelled images. This data therefore varies in scale, rotation and location. This similarity variation, must first be removed, by aligning the shapes into a common co-ordinate frame, before any analysis of intrinsic shape can take place.

5.3 Aligning the Training Data

To model the intrinsic shape variability all shapes must be aligned into a common co-ordinate frame, free of variation due to similarity transformations. This is achieved using Procrustes Analysis [34] which applies transforms to each shape in order to minimise the squared distance to the mean ($\sum |\mathbf{x}_i - \bar{\mathbf{x}}|^2$), where $\bar{\mathbf{x}}$ is the mean shape given a set of shapes $\{\mathbf{x}_i\}$

Firstly the shape data is reformatted to a more convenient notation. The x and y co-ordinates of each 2D point are concatenated to form a single vector, as shown in Equation 5.1.

$$\mathbf{x} = (x_1, \dots, x_n, y_1, \dots, y_n)^T \quad (5.1)$$

Given s training examples, s such vectors \mathbf{x} are generated. Each vector is of length $2n$. In this case $n = 20$ landmark points and $s = 1055$ training examples. Procrustes analysis then proceeds via the following iterative method, due to Cootes *et al.* [10], see Algorithm 5.1.

Algorithm 5.1 Aligning the Training Data [10]

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

The alignment procedure in step 4 of Algorithm 5.1 is critical to the final distribution of shapes. The transformations allowed when aligning each shape to the mean are scaling and orientation (shapes are already centred on the origin). However this step is followed by a projection into the “tangent space” [20] of the mean shape $\bar{\mathbf{x}}$. This is the hyperplane of vectors, normal to the mean shape, ie \mathbf{x}_t such that $(\mathbf{x}_t - \bar{\mathbf{x}}) \cdot \bar{\mathbf{x}} = 0$. The projection is achieved by scaling a given shape by $1/(\mathbf{x} \cdot \bar{\mathbf{x}})$. The tangent space projection improves the linearity of the aligned shape data.

A small training set of unaligned shapes is shown in Figure 5.2(a). The procedure described in algorithm 5.1 is then applied to produce the aligned shapes in Figure 5.2(b).

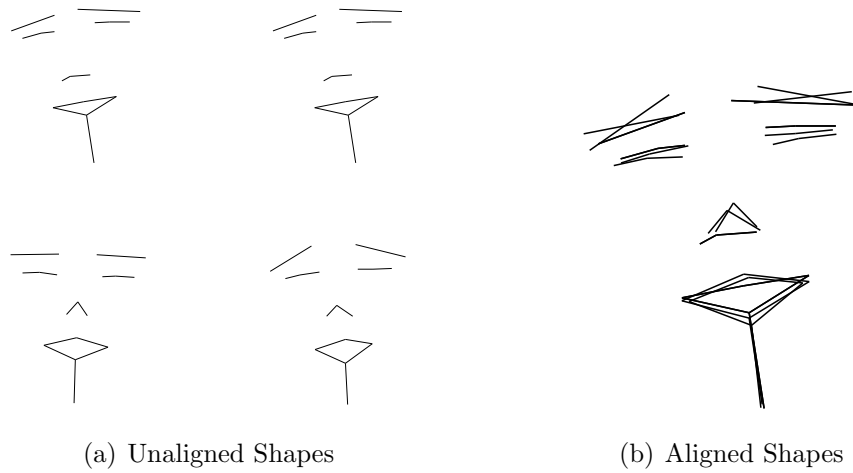


Figure 5.2: Aligning a set of shapes

5.4 Building a Shape Model

The alignment procedure described in Section 5.3 produces a set of s vectors \mathbf{x}_i , which form a distribution in a $2n$ dimensional space. In our case $n = 20$ and $s = 1055$, so we have a cloud of 1055 points within a 40 dimensional space, known as “shape space”, because each point represents an example shape, free of translation, rotation and scale variation.

We wish to model this distribution. However the high dimensionality (40D) of the space is prohibitive. The dimensionality is reduced by applying Principal Components Analysis (PCA) [44]. PCA computes the main axes of a cloud of points and allows each point to be approximated using a small number of parameters. The approach is described in Algorithm 5.2.

Algorithm 5.2 Principal Components Analysis

1. Compute the mean of the data $\bar{\mathbf{x}} = \frac{1}{s} \sum_{i=1}^s \mathbf{x}_i$
 2. Compute the covariance of the data $\mathbf{S} = \frac{1}{s-1} \sum_{i=1}^s (\mathbf{x}_i - \bar{\mathbf{x}})(\mathbf{x}_i - \bar{\mathbf{x}})^T$
 3. Compute the eigenvectors, ϕ_i and corresponding eigenvalues λ_i of \mathbf{S} (sorted so that $\lambda_i \geq \lambda_{i+1}$).
 4. Compute the sum of eigenvalues $V = \sum_{i=0}^n \lambda_i$ and find the smallest t such that $\sum_{i=0}^t \lambda_i \geq pV$, where p is manually defined.
 5. Store the first t eigenvectors as a matrix $\Phi = (\phi_1 | \phi_2 | \dots | \phi_t)$
-

The mean shape $\bar{\mathbf{x}}$ and matrix Φ define a linear subspace. A shape \mathbf{x} can be projected into the subspace, using Equation 5.2.

$$\mathbf{b} = \Phi^T(\mathbf{x} - \bar{\mathbf{x}}) \quad (5.2)$$

The set of parameters \mathbf{b} have dimension $t < n$ and can be used to approximate the original shape vectors \mathbf{x} (dimension n) as shown in Equation 5.3.

$$\mathbf{x} \approx \bar{\mathbf{x}} + \Phi \mathbf{b} \quad (5.3)$$

Following Dryden and Mardia [20], the individual parameters b_i are assumed to be independent and Gaussian. The variance across the training set of an individual parameter b_i is given by λ_i . Therefore the original shapes can be modelled by varying the parameters b_i .

By applying limits of $\pm 3\sqrt{\lambda_i}$ to the parameter b_i we ensure that the shape generated is similar to those in the original training set. The variation attributed to the first two modes of the human face shape model built are shown in Figure 5.3.

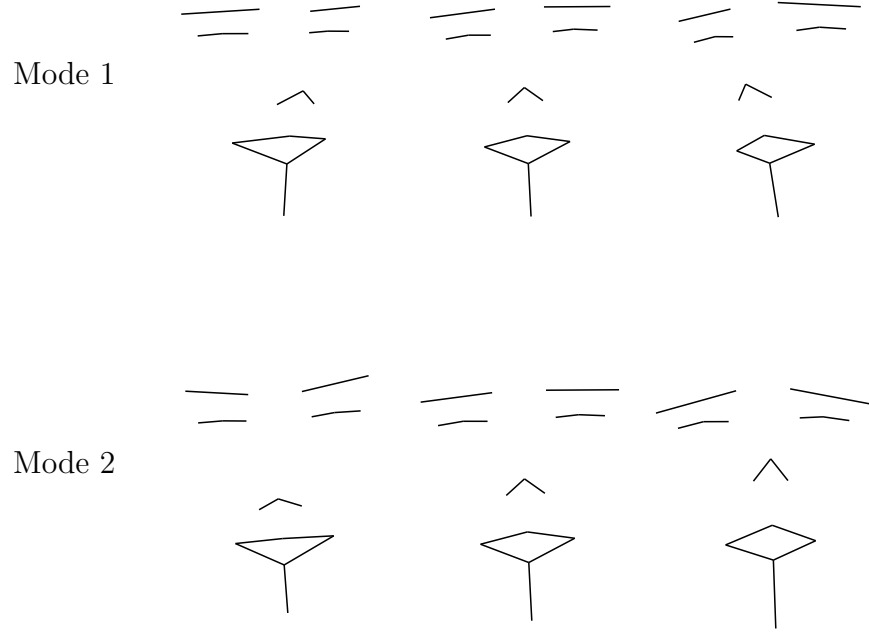


Figure 5.3: First two modes of the human face shape model

The dimensionality reduction due to PCA is controlled by the parameter p (see step 4 of algorithm 5.2). Here p is the proportion of the total variance retained. If p is small then $t \ll n$ and only a small number of eigenvectors are retained, but the PCA parameters may only provide a very rough approximation to the original shape vectors \mathbf{x}_j . If p is large then the original shape vectors will be more accurately represented, but more eigenvectors will be retained, so the dimensionality of the data may still be high. For example, setting $p = 0.9$ reduces the dimension of the PCA space to $t = 14$ compared to the original 40-D shape space.

5.5 Fitting to New Points

Given a new set of points \mathbf{Y} the shape model can be fitted to the new shape by finding a suitable transformation $T_{X_t, Y_t, s, \theta}$ and set of shape model parameters \mathbf{b} . The new set of points are approximated using Equation 5.4.

$$\mathbf{Y} \approx T_{X_t, Y_t, s, \theta}(\bar{\mathbf{x}} + \Phi \mathbf{b}) \quad (5.4)$$

Here the function $T_{X_t, Y_t, s, \theta}$ performs a rotation by θ , a scaling by s and a translation by (X_t, Y_t) . The transformation $T_{X_t, Y_t, s, \theta}$ and \mathbf{b} are chosen to minimise the sum of squares error.

$$|\mathbf{Y} - T_{X_t, Y_t, s, \theta}(\bar{\mathbf{x}} + \Phi \mathbf{b})|^2 \quad (5.5)$$

Cootes *et al.* [10] define an algorithm to iteratively minimise the approximation error (Equation 5.5, see Algorithm 5.3). In Algorithm 5.3 convergence is declared when applying an iteration produces no significant change in the pose or shape parameters. Converges usually takes only a few iterations.

5.5.1 Fitting to a Subset of Points

Frequently a shape model is required to fit to a subset of points and estimate the locations of the missing points. For example a shape model may be built from both eyes and both mouth corners, but be required to fit to a shape consisting of only one eye and both mouth corners.

Fitting to a subset of points is achieved using algorithm 5.3, however there are two changes to the algorithm:-

Algorithm 5.3 Iterative Fitting to New Points [10]

1. Initialise the shape parameters, \mathbf{b} , to zero
2. Generate the model instance $\mathbf{x} = \bar{\mathbf{x}} + \Phi\mathbf{b}$
3. Find the similarity transformation $T_{X_t, Y_t, s, \theta}$ which best maps \mathbf{x} to \mathbf{Y} (see Appendix A)
4. Invert the pose parameters and use to project \mathbf{Y} into the model co-ordinate frame:

$$\mathbf{y} = T_{X_t, Y_t, s, \theta}^{-1}(\mathbf{Y}) \quad (5.6)$$

5. Project \mathbf{y} into the tangent plane to $\bar{\mathbf{x}}$ by scaling by $1/(\mathbf{y} \cdot \bar{\mathbf{x}})$.
6. Update the model parameters to match to \mathbf{y}

$$\mathbf{b} = \Phi^T(\mathbf{y} - \bar{\mathbf{x}}) \quad (5.7)$$

7. Apply constraints on \mathbf{b}
 8. If not converged, return to step 2.
-

1. In step 3 the calculation of the transform $T_{X_t, Y_t, s, \theta}$ proceeds in the same manner, but only the subset of points are considered in the calculation.
2. In step 6 the parameters \mathbf{b} of the PCA space must be estimated by minimising the expression $|\mathbf{y} - \bar{\mathbf{x}} - \Phi\mathbf{b}|^2$, with reference to a weight vector specifying the relative importance of each model point. Here the weight vector has value one for points that are present and zero otherwise. The weighted fit procedure is described in Appendix B.

Using this scheme a shape model can fit to a subset of points and the shape model parameters found \mathbf{b} automatically suggest a location for the missing points, which can be calculated using Equation 5.4.

5.6 Likelihood of a Shape

Given an unseen shape \mathbf{Y} we would like to estimate the probability $p(\mathbf{Y})$ of the shape being a valid face. This is achieved by fitting the shape model to the face points (as described in Section 5.5) to extract the closest fitting model parameters \mathbf{b} and the projection of \mathbf{Y} in the aligned space \mathbf{x} . The residual error \mathbf{r} of the PCA model in the aligned space is defined as follows.

$$\begin{aligned}\mathbf{r} &= \mathbf{x} - \mathbf{x}' \\ \text{where } \mathbf{x}' &= \bar{\mathbf{x}} + \Phi \mathbf{b}\end{aligned}\tag{5.8}$$

The probability $p(\mathbf{Y})$ is analogous to the probability $p(\mathbf{x})$ in the aligned shape space. The shape \mathbf{x} is represented by a combination of the PCA parameters \mathbf{b} and the residual error \mathbf{r} . The distributions of \mathbf{r} and \mathbf{b} are assumed to be independent therefore

$$\log p(\mathbf{x}) = \log p(\mathbf{b}) + \log p(\mathbf{r})\tag{5.9}$$

Following Dryden and Mardia [20], the individual b_i parameters are assumed independent and gaussian. The log likelihood of a vector \mathbf{b} representing a plausible shape is therefore

$$\log p(\mathbf{b}) = -0.5 \sum_{i=1}^t \frac{b_i^2}{\lambda_i} + \text{const}\tag{5.10}$$

Here λ_i are the eigenvalues of each shape mode, which correspond to the variance of each parameter b_i , centred on zero and assumed gaussian.

Similarly it is assumed that each element of \mathbf{r} is independent and gaussian distributed with variance σ_r^2 , which can be estimated from the training set, so that

$$\log p(\mathbf{r}) = -0.5 \frac{|\mathbf{r}|^2}{\sigma_r^2} + \text{const} \quad (5.11)$$

The $p(\mathbf{x})$ can therefore be approximated using the following expression

$$\log p(\mathbf{x}) = -0.5 \left(\sum_{i=1}^t \frac{b_i^2}{\lambda_i} + \frac{|\mathbf{r}|^2}{\sigma_r^2} \right) \quad (5.12)$$

Given the log likelihood measure defined in Equation 5.12 a threshold p_t can be learnt from the training set on the quantity $\log p(\mathbf{x})$ such that a shape is defined plausible if $\log p(\mathbf{x}) > p_t$. Typically the threshold is chosen such that 98% of the training set will pass. Some examples of log likelihood scores for various shapes are shown in the Figure 5.4.

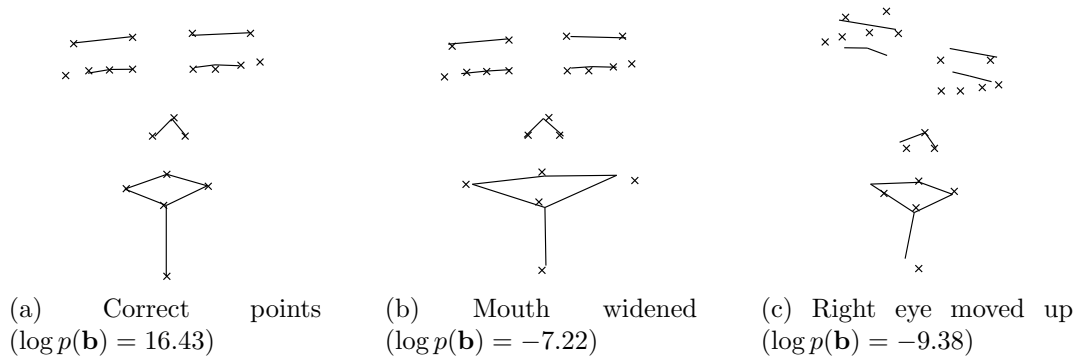


Figure 5.4: Example log likelihoods when fitting to 20 face points (denoted by x) after applying various distortions

In Figure 5.4 the fitted model is depicted using line segments. The points that are being tested are denoted by “x”. When fitting to correct points the log likelihood $\log p(\mathbf{b}) = 16.43$. When the correct points are distorted by widening the mouth to an artificially large width, the log likelihood of the 20pts representing a face drops to -7.22. When the right eye is raised to an unnaturally high level, relative to the left eye, $\log p(\mathbf{b})$ decreases to -9.38. This shows that the shape model can be used to

discriminate between plausible and non-plausible face point configurations.

Sometimes only a subset of face points are available. For example the eye pupils and the mouth corners. However, the full shape model can still be fitted to this four point subset using the method described in Section 5.5.1. Figure 5.5, shows examples of fitting to shapes using only the eye pupils and mouth corners.

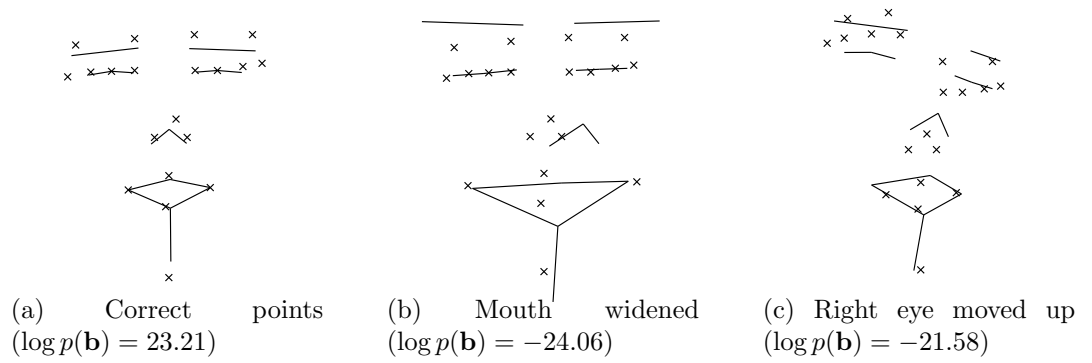


Figure 5.5: Example log likelihoods when fitting to 4 face points, (both pupils and both mouth corners) after applying various distortions

There are some numerical changes in the log likelihood of each shape $\log p(\mathbf{b})$ when considering only a four point subset. The likelihood of the the correct points has increased $16.43 \rightarrow 23.21$. This is due to the shape model being able to fit to fewer points and therefore produce a fitted shape closer to the mean of the shape model, therefore $\log p(\mathbf{b})$ increases. However, the likelihood of the mouth distorted shape has decreased $-7.22 \rightarrow -24.06$, similarly the likelihood of the eye distorted shape has decreased $-9.38 \rightarrow -21.58$. This is due to the shape fitting to the distorted points and ignoring the influence of the other correct points, resulting in a more unlikely model parameters. However, despite these changes in the magnitude of $\log p(\mathbf{b})$ the shape model is still able to discriminate between the correct points and distorted points when fitting to the eye pupils and the mouth corners.

5.7 Conclusions

This chapter has shown that the shape variation of 20 landmark points over the 1055 WEBCAM images can be modelled by aligning the shape data and projecting into a 14 parameter PCA space. Examples are also shown of the shape model fitted to valid face points and invalid face shapes. The model can be used to discriminate between plausible and non-plausible configurations of points, assuming a Gaussian distribution of the shape model parameters in the PCA space. This form of statistical shape model is used as a discriminator in the next chapter, which aims to constrain the search for individual facial features.

Chapter 6

Shape Constrained Feature Detection

This chapter describes three methods of combining shape and feature detection to accurately locate facial features. The methods perform a local search given the approximate location and orientation of the face. The Boosted Cascade Face Detector described in Section 3.5 and Chapter 4 is used to locate upright faces and the techniques described below are used to find individual facial features, such as the eye pupils and corners of the mouth.

6.1 Local Feature Detectors

The Boosted Cascade Face Detector returns a region which is assumed to contain a face. Here, a search is deemed successful if $d_{eyes} < 0.3$, following Section 3.7.2. Given a successful global match to an unseen image, local feature detectors are applied to search for individual facial features. The type of feature detector used may be any form of 2D pattern recognition method. In this chapter, the three methods used to model the whole face in Chapter 3, are now used to model local image patches

centred on individual facial features.

6.1.1 Training Data

The training set is the WEBCAM data set described in Section 3.1. An example training image from the WEBCAM data set is shown in Figure 6.1(a).

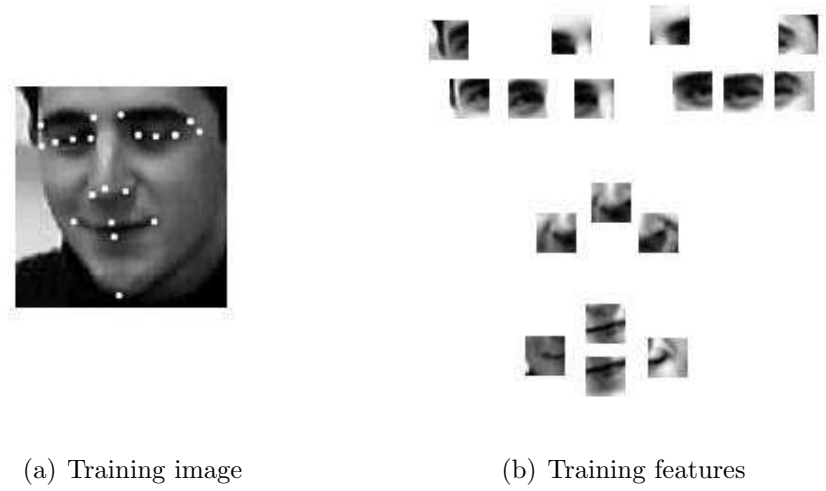


Figure 6.1: Example feature training images

There are 17 internal landmarks on each training image (the temples and the chin are excluded). A region with dimensions proportional to the inter-ocular separation of the training face, is formed around each feature point and the feature patch subsampled to 15x15 pixels. Examples of subsampled patches are shown in Figure 6.1(b) and further examples of data used to train the 17 feature models are shown in appendix D.

6.1.2 Individual Feature Detectors

Given the training patches described in Section 6.1.1 local detectors can be built for each facial feature. Similar to Chapter 3, there are three different types of feature detector.

1. Normalised Correlation Detector (NCD) - The average training image for each detector is scaled such that the pixel values have zero mean and unit variance, as described in Section 3.4. The templates for each feature are shown in Appendix E.
2. Orientation Map Detector (OMD) - Orientation maps are constructed by application of a Söbel edge filter, to each averaged feature image, using the method described by Fröba and Küllbeck [30] and previously applied to whole face detection in Section 3.3.1. The orientation maps for all 17 features are shown in Appendix F.
3. Boosted Cascade Detector (BCD) - The training set described in Section 6.1.1 is used to build a 10 level cascade containing 10 features in each level, using the method described in Section 3.5. The first six features selected by AdaBoost for each facial feature are shown in appendix G.

6.1.3 Unconstrained Feature Search

Given a set of individual feature detectors, feature search proceeds by first applying the global face detector to an image containing a human face. Assuming global search is successful, the whole face region is used to predict a search region for each individual facial feature. The bounding box for each feature is precomputed by applying the global detector to a verification set. A feature detector is then applied to each region and the best match of each detector recorded as the predicted location of each feature. The search is unconstrained, in the sense that each local detector is independent, so the final configuration of points is not guaranteed to form a plausible face shape.

6.1.4 Proximity Measure

To assess the accuracy of feature detection the locations found by each feature detector are compared with manually labelled feature points. The average point to point error (m_e) is calculated as follows.

$$m_e = \frac{1}{ns} \sum_{i=1}^{i=n} d_i \quad (6.1)$$

Where d_i are the point to point errors and s is the inter-ocular distance as shown in Figure 6.2. Here n is the number of feature points modelled.

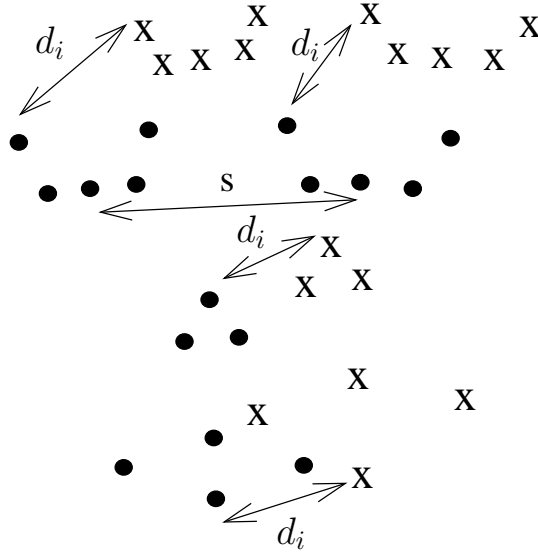


Figure 6.2: Proximity Measure, “.” = true feature location and “x” = predicted feature location

The mean error (m_e) used to assess search accuracy can be computed over all feature points when it is referred to as m_{e17} . However, the mean error is sometimes restricted to just four points, i.e. the eye pupils and mouth corners, when it is referred to as m_{e4} . In Section 3.7.1 the d_{eyes} metric, used to assess face detection performance, can be thought of as m_{e2} , where the mean error is computed using the eye pupils only.

Similarly m_{e1} is sometimes used to denote the error of an individual feature point.

6.1.5 Unconstrained Search Results

Unconstrained feature search is applied to the BIODID subset images described in Section 3.6.2. Figure 6.3 shows the accuracy of unconstrained feature search relative to manually labelled points. Various proximity measures are used. For example, Figure 6.3(a) plots the proportion of successful searches using the mean error over all 17 feature points (m_{e17}). Whilst Figures 6.3(b) , 6.3(c) and 6.3(d) plot the mean error for the right eye, left mouth corner and nose tip feature detectors only.

The feature points can also be predicted by simply computing the mean feature locations from the global face region, without performing any local search. For comparison purposes the average predicted point accuracies are also plotted with the local search results in Figure 6.3.

Figure 6.3(a) shows that using the mean error (m_{e17}) over all feature points, all three local detection methods give worse performance than merely computing the average feature locations from the global face detection. For example, if the required accuracy is $m_{e17} < 0.15$ then the proportion of successful searches using the average points predicted by the global candidate is 85%. Using unconstrained BCDs the success rate drops to 75%. Whilst with normalised correlation, performance drops dramatically to only 22% success rate and 9% with orientation maps.

Figures 6.3(b) , 6.3(c) and 6.3(d) show that the search accuracy varies from feature to feature. For example, the right eye detector is often much more accurate than the average right eye prediction using the global candidate region (see Figure 6.3(b)). With Boosted Cascade Detectors the right eye detector has an accuracy of $m_{e1} < 0.05$ in 80% of cases, this drops to 25% with average point prediction. Figures 6.3(c) and 6.3(d) also show that the mouth corner and nose tip can sometimes be located more accurately with local feature detectors instead of the average points.

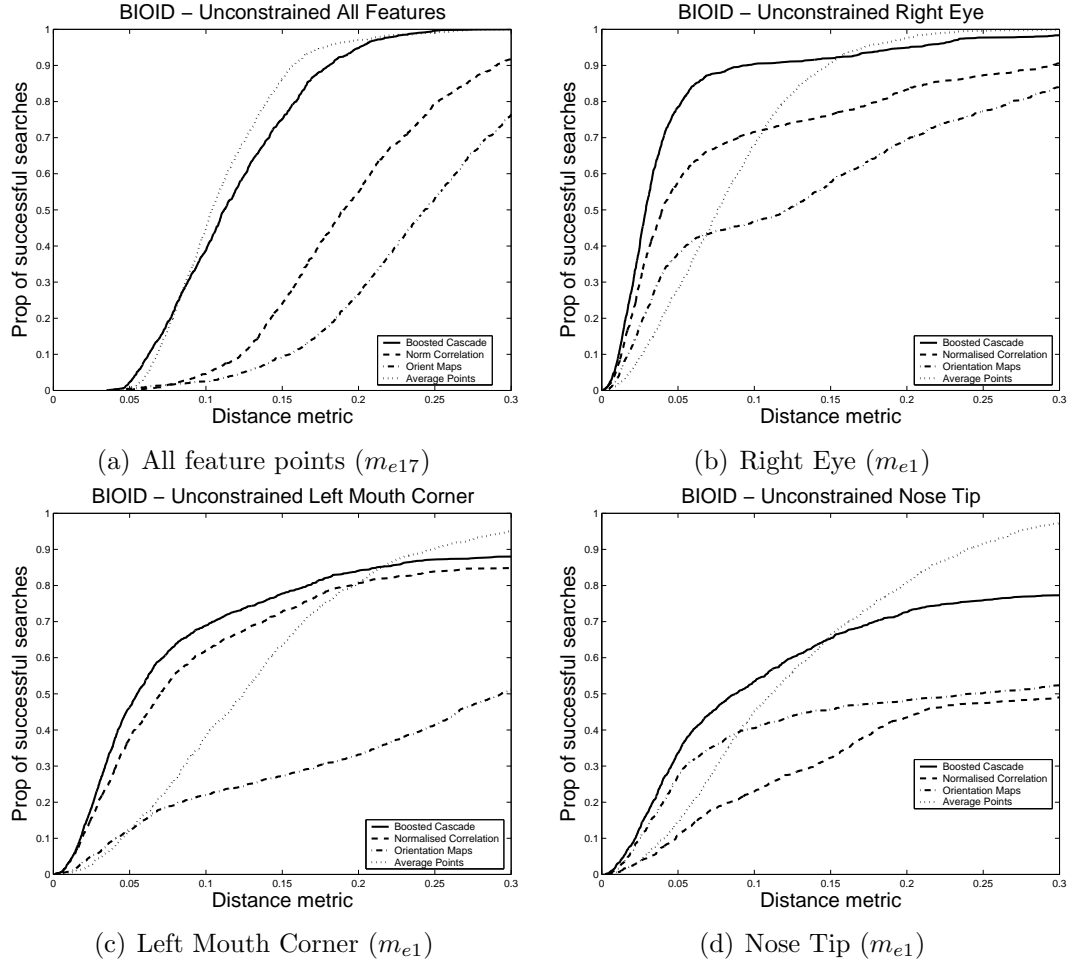


Figure 6.3: Point to point error when performing unconstrained feature detection on the BIOD test set

However, the local feature search is often unsuccessful. Especially when applied to the left mouth corner and nose tip. When the local search fails the point to point error can be very large. Therefore the average error (m_{e17}) over all features is often worse than the average error for the mean points (see Figure 6.3(a)).

Figure 6.3 shows that generally the Boosted Cascade Detectors give better performance than the Orientation Map Detectors and Normalised Correlation Detectors. This is consistent with Chapter 3, which showed that the Boosted Cascade Detector is also the most successful face detector. However, unconstrained Boosted Cascade Detector search is still less reliable than the average point locations using the m_{e17} distance measure.

6.1.6 Unconstrained Search Examples

Examples of unconstrained search using Boosted Cascade Detectors are shown in Figure 6.4.

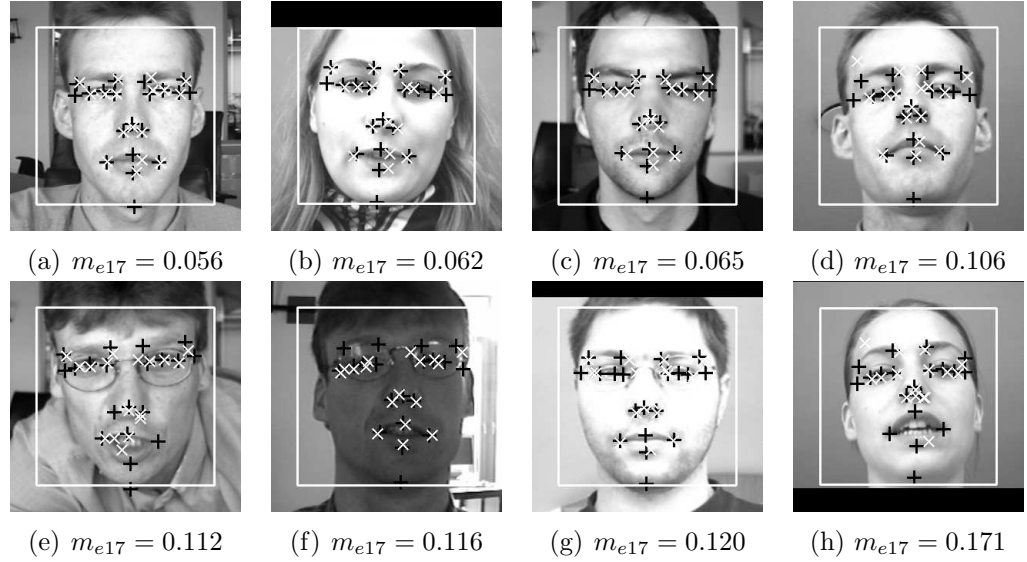


Figure 6.4: Examples of unconstrained search with Boosted Cascade Detectors

In Figures 6.4(a) , 6.4(b) and 6.4(c) the unconstrained search works reasonably well. However in the other examples the unconstrained search finds some features, but fails spectacularly for other features. For example, the outer right brow corner detector fails in Figure 6.4(d). The mouth detectors fail in Figure 6.4(e). The eye region detectors fail in Figures 6.4(f) and 6.4(g). Multiple feature detectors fail in Figure 6.4(h).

In Figure 6.4, the individual feature searches are independent, so in many cases, the false feature detections fail to form a plausible face shape. Constraining feature detections using shape is explored later in this chapter.

6.1.7 Unconstrained Search Timing

Time trials were carried out on the different unconstrained feature detectors on a PII 500Mhz PC. The results are summarised in Table 6.1.

Detector	Time for local feature search (17pts)
boosted cascade	~150ms
normalised correlation	~300ms
orientation map	~550ms

Table 6.1: Time to perform unconstrained feature search on a single BIOID image using a 500Mhz PII processor

Table 6.1 shows that the BCD is the quickest local feature detector. This is because the efficient cascade structure discards unlikely regions very quickly. In Section 3.2 of Chapter 3, the normalised correlation and orientation map face detectors are actually quicker than the Boosted Cascade Detector at searching the whole image. However, the reason for this is the efficient coarse-to-fine mechanism described in Section 3.3.4, which is only used for global face search.

6.1.8 Unconstrained Search Conclusions

The large point to point errors in Figure 6.3 and poor results shown in Figure 6.4 show that purely local search is unreliable. The individual feature detectors described in this chapter cannot improve on the accuracy of the average feature locations predicted using the global candidate. This is probably due to individual features having a high variability in appearance (see feature training examples in Appendix D) and therefore not having enough fixed local structure to enable reliable detection.

In order to improve the accuracy of local search the author has developed and implemented several methods of constraining local feature detectors by employing shape constraints on the configuration of points suggested by the individual feature detectors. Three such methods of combining shape and feature detection are implemented

and tested. Combinatoric Shape Search (CSS) is described in Section 6.2. Shape Optimised Search (SOS) is described in Section 6.3. Pairwise Reinforcement of Feature Responses (PRFR) is described in Section 6.4.

6.2 Combinatoric Shape Search (CSS)

This method of constraining feature detectors was first implemented by the author and described in Cristinacce and Cootes [15]. The method proceeds by retaining the best feature responses for each detector. All combinations of feature locations are then grouped to form point sets. The set of most likely feature points that also passes a threshold on the shape likelihood is deemed the final output of the algorithm.

The expectation is that the shape constraint will avoid invalid matches such as the feature detections depicted in Figure 6.4. The statistical shape model, used to test point configurations, is constructed as described in Chapter 5. The individual feature detectors may be any of the detectors described in Section 6.1. Additionally, the probability of an individual feature detector finding the correct location is estimated, as described below.

6.2.1 Probability of Individual Feature Responses

The probability of a correct match for an individual feature detector is approximated, by learning the distribution of responses to both positive and negative feature examples. Given $P(r|F)^*$ the probability of a given response at the true feature location and $P(r|B)$ the probability of a given response away from the true feature location, the probability of a given feature response, r , matching the correct point within the search window, $P(F|r)$, is calculated using Bayes Theorem (see Equation 6.2).

*In practice r is a continuous variable, therefore this is the probability of r lying within some interval

$$P(F|r) = \frac{P(r|F)P(F)}{P(r|F)P(F) + (1-P(F))P(r|B)} \quad (6.2)$$

Here $P(r|F)$ is estimated by calculating a histogram of responses to a positive verification set and $P(r|B)$ is similarly estimated using a histogram calculated using a negative verification set. $P(F)$ is the prior probability of a correct feature match and is estimated as $P(F) = 1/n$, where n is the number of points evaluated by the feature detector within the search region.

For each feature detector a suitable threshold T_f is learnt from a verification set, such that 90% of true feature patches pass the threshold. Therefore a feature detection is only accepted if $P(F|r) > T_f$, otherwise it is discarded.

6.2.2 Shape Constraints

The feature detections described in Section 6.2.1 are combined into candidate point sets. Each candidate point set contains at most one candidate location for each feature detector. A candidate point set must contain at least $k > 2$ detected features, with the remaining $(n-k)$ points predicted by fitting a shape model to the detected points.

To be considered as a possible face match, a candidate set of points must pass two types of shape constraint as follows.

1. A shape model is fitted to the set of points. The likelihood of the shape $p_s(\mathbf{x})$, is estimated using the the method described in Section 5.6. If $p_s(\mathbf{x}) > T_s$, the shape is accepted, otherwise it is discarded.
2. Limits are placed on the transformation between the candidate point set and the average points predicted by the global candidate. The maximum scale variation is s_{max} , the max rotation is θ_{max} and the max translations x_{max} and y_{max} .

Here the transformation limits are set with knowledge of the accuracy of the global

search. In this thesis the limits used are $s_{max} = 10\%$, $\theta_{max} = 10^\circ$ and $x_{max} = y_{max} = 10\%$. The threshold T_s is set such that 90% of the training set shapes would pass the threshold.

6.2.3 Point Set Objective Function

Given a list of candidate point sets $\{\mathbf{X}\}$, which pass the shape constraints described in Section 6.2.2, the objective function $S(\mathbf{X})$ is used to select the best point set, as follows.

$$S(\mathbf{X}) = \sum_{i=1}^{i=k} P_i(F|r) + k \quad (6.3)$$

Where k is the number of feature detections in the point set and $P_i(F|r)$ is the probability of a correct match for each detected feature point, as described in Section 6.2.1. This objective function is designed to ensure that a candidate set with 4 points is always ranked higher than a candidate set with only 3 points.

6.2.4 Finding the Best Candidate Point Set

Given a list of feature responses for each detector, a simple method of selecting the best candidate point set is to form all possible candidates and select the highest scoring point set \mathbf{X} which passes the constraints imposed by the shape model. However, an obvious problem with this approach is the combinatorial explosion that results if too many candidate points are returned for too many feature points. For example with 7 feature points each returning 9 candidate locations and allowing for missing features, the number of possible face candidates is $(9 + 1)^7 = 10^7$. It is infeasible to evaluate this number of candidates in a sensible amount of time.

We wish to find the best candidate point set, using the feature responses and the

shape constraints, but also limit the combinatorial explosion. This is achieved in three ways, as follows.

1. Limit n_p the number of feature points considered
2. Restrict the number of points any one feature detector can return n_{max}
3. Employ an efficient iterative search method, as described in algorithm 6.1.

Algorithm 6.1 Iterative Search Method

1. Order the feature candidates for each feature detector, largest response first.
 2. Set $i = 1$
 3. Restrict the search for a face candidate to the best i responses of each detector.
 4. If a feature detector has less than i point candidates then it is allowed a wildcard.
 5. Ensure that each face candidate contains at least one candidate point that is the i^{th} response of a detector [†]
 6. Form all face candidates and test each with the shape model.
 7. If no face candidate passes the shape constraint set $i = i + 1$ and go to step 3.
 8. Rank all face candidates that pass the shape constraint using the objective function described in Equation 6.3
 9. If the best candidate has missing features these are predicted using the shape model.
-

The aim of algorithm 6.1 is to select the highest scoring candidate with the maximum number of feature points that satisfy the shape constraint. The procedure makes searching tractable by ignoring weaker point candidates and returns the same candidates as a full search.

The main parameters used in algorithm 6.1 are the number of feature points considered n_p and the maximum number of candidate points retained by each feature

[†]This ensures that point sets from previous iterations of i are not re-evaluated

detector n_{max} . In this paper, $n_p = 4$ and the features modelled are the centre of the pupils and the corners of the mouth, while $n_{max}=5$.

These parameters mean that a point set must be found from the top five point candidate returned by each feature detector. If a feature detector returns less than 5 point candidates, because the probability of any correct match is too low, then a wildcard will eventually be used for that feature. In practice these values for n_p and n_{max} give good results and reduce the processing time relative to a full search of all possible point candidates.

6.2.5 CSS Results

The effectiveness of the Combinatoric Shape Search algorithm is shown by applying the method to the BIODID test set. CSS is compared with unconstrained feature search and average point prediction by computing the mean point to point error (m_{e4}). Here m_{e4} is the average mean error of the eye pupils and mouth corner points, as described in Section 6.1.4. Figure 6.5 plots the proportion of successful searches given various thresholds of m_{e4} , using CSS and a variety of feature detectors.

Figure 6.5(a) shows that using CSS with Boosted Cascade Detectors provides more accurate feature locations compared to the average points predicted by the global face candidate. With a required accuracy of $m_{e4} < 0.15$, the average point prediction is successful for 83% of faces. However, when using Boosted Cascade Detectors and Combinatoric Shape Search the number of successful searches is 90%. Using CSS the proportion of successful searches is greater at all values of m_{e4} . Therefore CSS is a clear improvement over the average points predicted from the global candidate, when using Boosted Cascade Detectors.

Figures 6.5(b) and 6.5(c) show that when using orientation maps or normalised correlation with CSS the performance is only similar to the average point prediction. For example with $m_e < 0.15$, Orientation Map Detectors are successful for 77% of

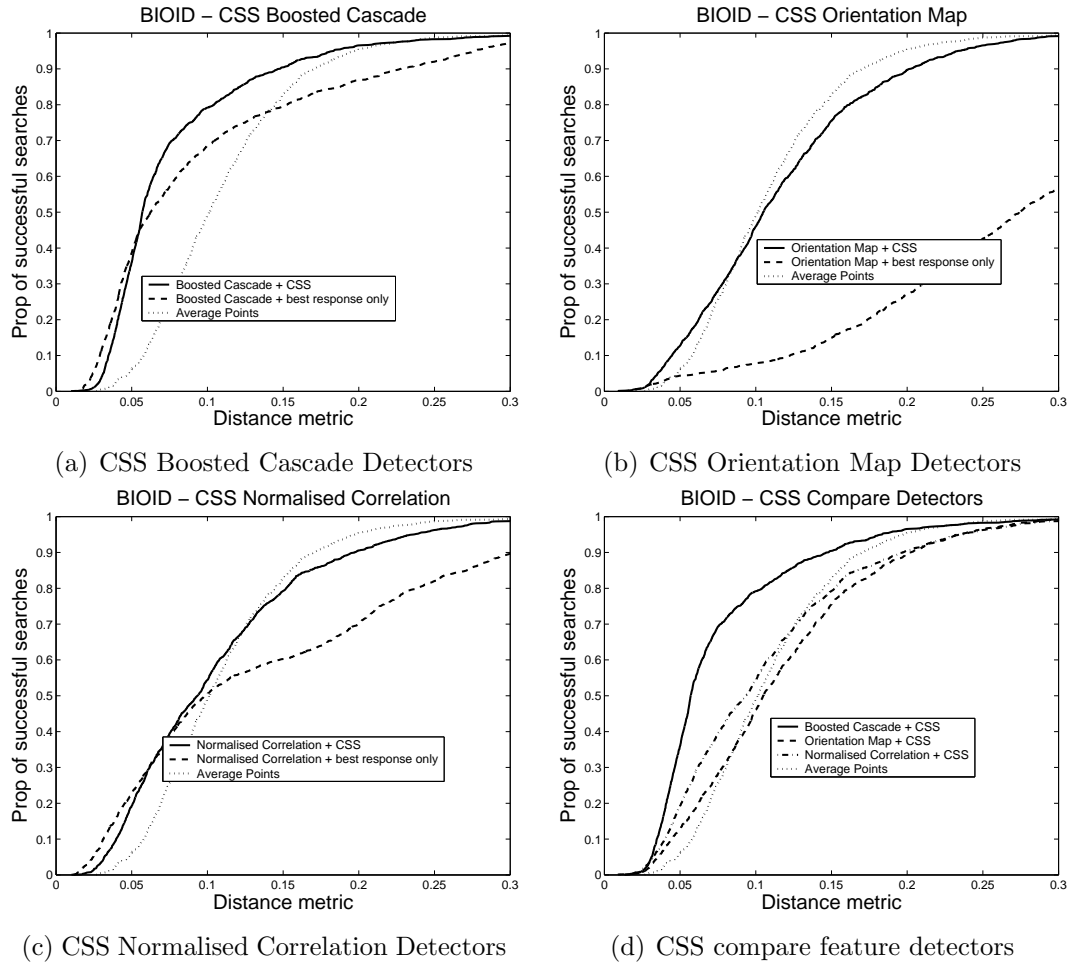


Figure 6.5: Average mean positional error (m_{e4}) when performing CSS on the BIOID test set

faces and Normalised Correlation Detectors 78% compared to 83% with average point prediction. However the Combinatoric Shape Search algorithm does give a vast improvement on unconstrained search. For normalised correlation with $m_e < 0.15$ the success rate improves 59% \rightarrow 78% (Figure 6.5(c)). Whilst for orientation maps the success rate improves 15% \rightarrow 77% (Figure 6.5(b)). Therefore, Combinatoric Shape Search gives a vast improvement on unconstrained search for both the Normalised Correlation Detector and Orientation Map Detector, however the end results are not much better than average point prediction.

Figure 6.5(d) confirms that Boosted Cascade Detectors with Combinatoric Shape Search is by far the best method, performing better than Combinatoric Shape Search

with normalised correlation or orientation maps. Therefore given a reliable enough detector Combinatoric Shape Search makes the local search more robust, when searching for the eye pupils and mouth corners.

6.2.6 CSS Example Searches

The effectiveness of the CSS algorithm combined with Boosted Cascade Detectors is illustrated by comparing examples of CSS search with unconstrained Boosted Cascade Detector search. Figure 6.6 shows examples of constrained and unconstrained searches.

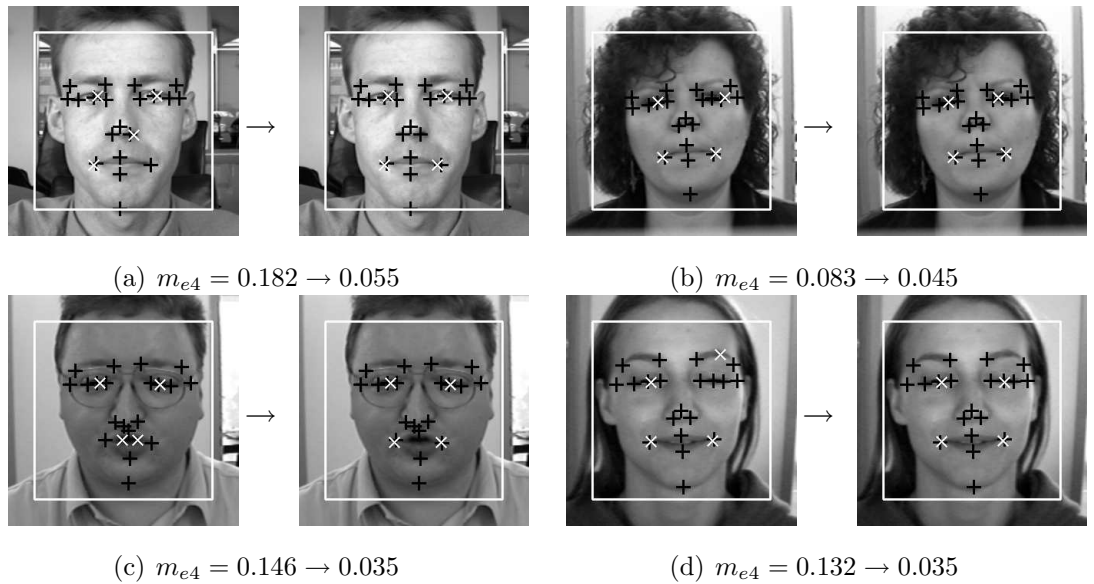


Figure 6.6: Examples of improvement in search accuracy (m_{e4}) when using CSS compared to unconstrained search, using Boosted Cascade Detectors

In Figure 6.6(a) the unconstrained search falsely matches the left mouth corner detector to the left nostril. However, this point configuration does not pass the shape model constraint. The CSS algorithm therefore corrects the false detection and improves the mean error (m_{e4}) from $0.182 \rightarrow 0.055$. Similar improvements are shown in Figure 6.6(b), where a false detection to the left eye corner is avoided, and Fig-

ure 6.6(d) where a false match to the left eye brow is corrected. In Figure 6.6(c), both predictions for the mouth corners are improved.

6.2.7 CSS Timing

Timing evaluations were carried out on the CSS algorithm, using Boosted Cascade Detectors, on a PII 500Mhz PC. The results are summarised in Table 6.2.

Event	Time
Global search	~400ms
Local feature search (4pts)	~50ms
Find best point candidate set	~0-150ms
Total	~450-600ms

Table 6.2: Time to perform CSS with BCD detectors on a single BIOID image using a 500Mhz PII processor

In Table 6.2, the time to find the best point set varies because sometime the best feature responses form a valid shape, but often this is not the case and more points sets have to be considered, increasing the search time. However, the whole search, i.e. global search and four point feature search can be completed in ~450-600ms even on a relatively old machine, so the CSS algorithm is efficient and would be able to search many times a second on more modern hardware.

6.2.8 CSS Conclusions

The CSS algorithm when combined with Boosted Cascade Detectors has been shown to locate feature points more accurately than average point predictions (see Figure 6.5(a)). The improvement in performance relative to unconstrained search, is due to the removal of false detections, which are deemed unlikely by the CSS shape model (see Figure 6.6 for example search results). The CSS method is shown to be efficient, allowing features to be located in ~450-600ms on a relatively old PC.

However, CSS is shown to be unreliable with weaker detectors, such as Normalised Correlation Detectors and Orientation Map Detectors. Therefore a reasonably reliable detector, such as the Boosted Cascade Detector is required for the CSS algorithm to work effectively.

A more serious problem with the CSS algorithm is that the combinatorial problems described in Section 6.2.4 restrict the number of feature points that can be modelled. This means that the shape constraints are weaker than they would be if more feature points were modelled. Also the locations of only four features are predicted by the algorithm. A multi-stage procedure would be required to predict more points. Therefore a different shape constraint method has been devised that can be more easily extended to a greater number of feature points. This method is described in Section 6.3 below.

6.3 Shape Optimised Search (SOS)

The Shape Optimised Search (SOS) method also attempts to combine feature detection with shape constraints. However, unlike CSS explicit point configurations are not considered. Instead the shape constraint is incorporated into an optimisation scheme. The SOS method is described below, but was originally described in Cristinacce and Cootes [16].

6.3.1 Feature Response Images

Instead of simply recording the highest ranking feature responses, the feature detectors are applied to their individual search regions and a quality of fit measure returned for each pixel within the region. Responses for the eye and mouth corner using Boosted Cascade Detectors are shown in Figure 6.7.

Dark regions in Figure 6.7 indicate a high likelihood of a correct match, whereas

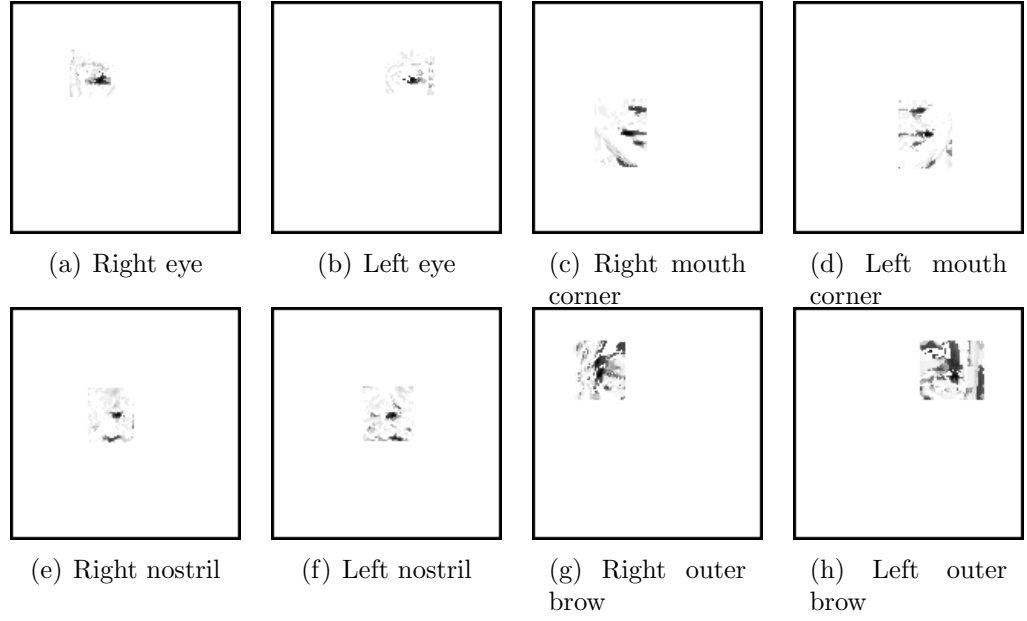


Figure 6.7: Response images for the eyes and mouth corner detectors (black implies strong response)

lighter regions indicate that a pixel is unlikely to be the correct feature location.

6.3.2 Non Linear Optimisation

Given a set of feature response images $\{I_i\}$, the mean feature locations within the object are used to initialise the shape. Here shape is represented as a concatenated vector of X and Y co-ordinates, as follows.

$$\mathbf{X} = (X_1, \dots, X_n, Y_1, \dots, Y_n)^T \quad (6.4)$$

The shape \mathbf{X} is represented by the shape parameters \mathbf{b} from Equation 5.3 and a transformation T_t from the shape model frame to the image frame. Therefore \mathbf{X} is calculated from the shape parameters as follows.

$$\mathbf{X} \approx T_t(\bar{\mathbf{x}} + \Phi\mathbf{b}) \quad (6.5)$$

Here $T_{\mathbf{t}}(x)$ applies a similarity transform with parameters \mathbf{t} . We concatenate the shape and pose parameters into $\mathbf{p} = (\mathbf{t}^T | \mathbf{b}^T)^T$. The objective function $f(\mathbf{p})$ for a given vector \mathbf{p} is then defined as follows.

$$f(\mathbf{p}) = \begin{cases} \sum_{i=1}^n I_i(X_i, Y_i) & \text{if } |b_i| < 3\sqrt{\lambda_i} \ \forall i \\ 0 & \text{otherwise} \end{cases} \quad (6.6)$$

Bilinear interpolation of the feature response image $I_i(X, Y)$ is used for non-integer (X, Y) .

The search algorithm is simply to vary the parameter vector \mathbf{p} to maximise the fitting function $f(\mathbf{p})$. This can be achieved using any optimisation technique. Throughout this thesis we use the Nelder-Meade simplex method [64]. This requires an initial step size to guide the optimisation of each element of vector \mathbf{p} . For each element b_i the initial step size is $\sqrt{\lambda_i}$. The initial step size for the pose elements t_i are such that the translation varies by 10% of the object size and the orientation and scale may vary by up to 20%. When the simplex method produces no significant change in the parameter vector \mathbf{p} , i.e. a maxima has been found, the algorithm terminates and the final shape parameters \mathbf{p} determine the location of each feature.

6.3.3 SOS Results

Figure 6.8(a) shows that when using BCDs the SOS algorithm modelling 17 points outperforms the CSS method modelling 4 feature points, using the m_{e4} distance measure (see Section 6.1.4). However, the CSS method still outperforms the SOS algorithm when it is restricted to modelling the same four feature points as the CSS algorithm (i.e. the eye pupils and mouth corners).

For example in Figure 6.8(a) with a threshold of $m_{e4} < 0.15$, the SOS 17pt method is successful for 94% of faces. This compares to 90% for the CSS 4pt method. The SOS

17pt method is more successful at all possible values of m_{e4} , so is clearly superior. However, when restricted to four points SOS success rate drops to only 83% of faces, which is the same as average point prediction. Therefore with BCDs, SOS is much improved when modelling 17 feature points, compared to 4 feature points.

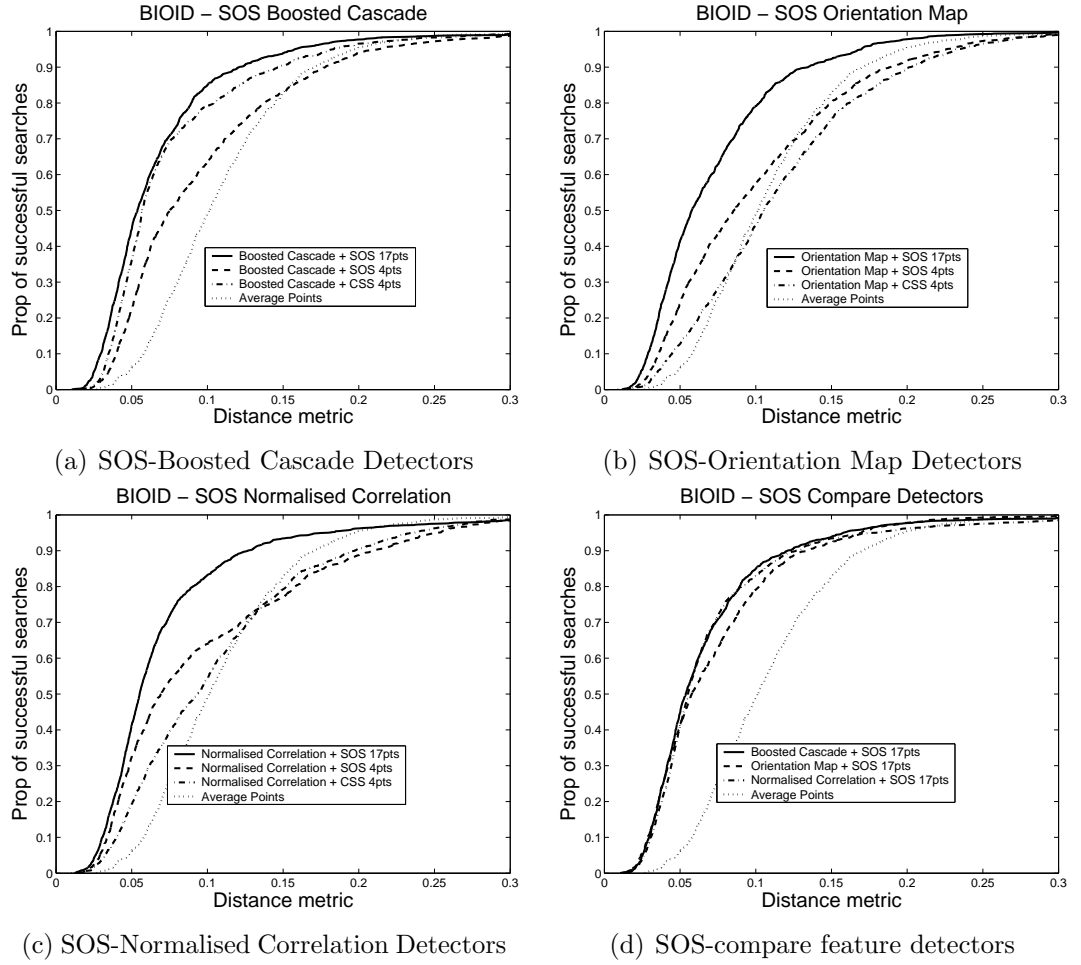


Figure 6.8: Average mean positional error (m_{e4}) when performing SOS on the BIOD test set[§]

Figures 6.8(b) and 6.8(c) show that when using orientation maps or normalised correlation SOS with 17 feature points works much better than any other method. Using OMD or NCD the CSS algorithm fails to improve on the average point predictions. However using SOS the improvement relative to the average points (with $m_{e4} < 0.15$)

[§]Note that in Figure 6.8 all point to point errors m_{e4} are calculated only using the eye pupil and mouth corners, irrespective of the other feature points that are modelled by the SOS algorithm.

is $83\% \rightarrow 93\%$ using orientation maps and $83\% \rightarrow 94\%$ using normalised correlation. Therefore the SOS algorithm works even with individually poor performing detectors such as OMDs and NCDs. However when only using 4 feature point SOS performance is much worse. Therefore the SOS algorithm needs many feature points to be effective.

Figure 6.8(d) confirms that the SOS 17pt algorithm gives similar performance, with all three feature detectors. Given a threshold of $m_{e4} < 0.15$ the Boosted Cascade Detector is successful for 94% of faces, the Normalised Correlation Detector 94% and the Orientation Map Detector 93%, compared to the average point prediction success rate of 83%. Therefore SOS with 17 feature points is a reliable and robust method even with relatively weak individual feature detectors.

6.3.4 SOS Search Examples

Examples of the SOS 17pt search compared with the CSS 4pt search using Boosted Cascade Detectors are shown in Figure 6.9.

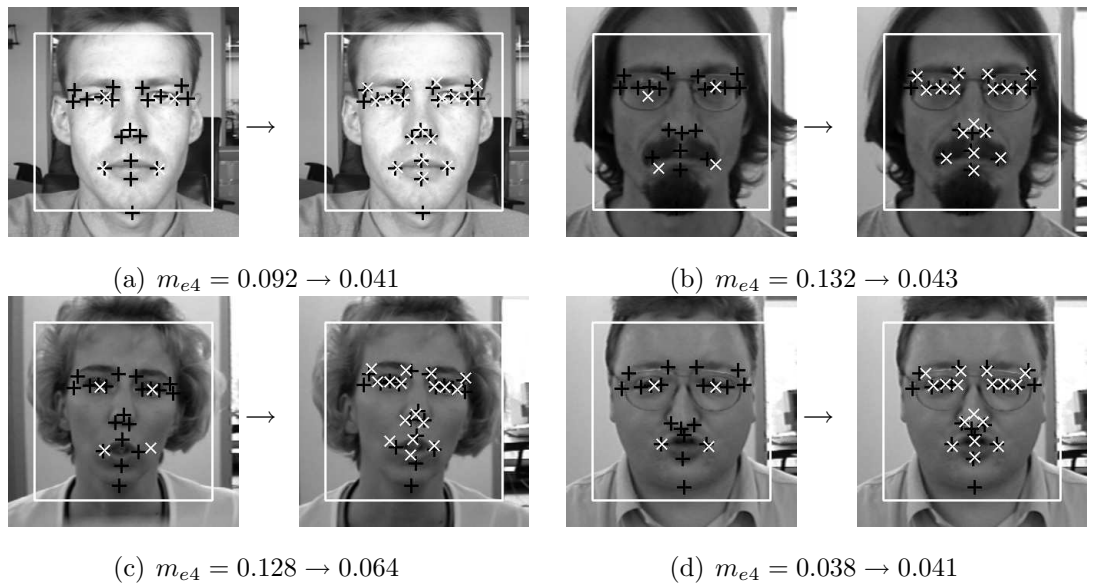


Figure 6.9: Examples of improvement in search accuracy (m_{e4}) using SOS 17pt search compared to CSS 4pt search with Boosted Cascade Detectors

Figure 6.9 shows improvements in search accuracy (m_{e4}) when using SOS 17pt search instead of CSS 4pt search. For example in Figure 6.9(a) the CSS search does not find the pupils accurately. The SOS algorithm is able to correct this and reduce the mean error (m_{e4}). Figures 6.9(b) and 6.9(c) show an improvement in mouth corner location using the SOS 17pt instead of the CSS 4pt method. Figure 6.9(d) shows that both methods give similar accuracy in a case where the CSS already performs well.

6.3.5 SOS Timing

The speed of the SOS algorithm using Boosted Cascade Detectors is evaluated on a PII 500Mhz PC. The results are summarised in Table 6.3.

Event	SOS 4pt	SOS 17pt
Global search	$\sim 400\text{ms}$	$\sim 400\text{ms}$
Local feature search	$\sim 50\text{ms}$	$\sim 150\text{ms}$
Time to optimise shape parameters	$\sim 0\text{-}50\text{ms}$	$\sim 0\text{-}200\text{ms}$
Total	$\sim 450\text{-}500\text{ms}$	$\sim 550\text{-}750\text{ms}$

Table 6.3: Time to perform SOS with BCDs on a single BIOID image using a 500Mhz PII processor

Table 6.3 shows that the global detector and SOS 17pt local search can find features in $\sim 550\text{-}750\text{ms}$. With four points the SOS finds features in $\sim 450\text{-}500\text{ms}$. however as Figure 6.8(a) shows, performance is dramatically reduced when only modelling four points.

The most variable step is the time taken for the Nelder-Meade downhill simplex to converge. The maximum observed optimisation time increases from $\sim 50\text{ms} \rightarrow \sim 200\text{ms}$ when moving from $4 \rightarrow 17$ points. The time to optimise the shape parameters is also highly variable between images. However even on a relatively old machine the SOS can find 17 facial features in less than a second.

6.3.6 SOS Conclusions

The SOS 17pt algorithm is shown to predict the eye pupil and mouth corner locations more accurately than the CSS 4pt algorithm. Figure 6.8 shows that the SOS 17pt method gives good performance with all feature detectors, even the weaker NCD + OMD detectors, which give very poor unconstrained search results. The SOS algorithm is efficient - able to find features in ~ 550 - 750 ms even on an old CPU.

The strength of the SOS algorithm compared to the CSS method is probably due to its ability to model many feature points instead of just four. With just four feature points the SOS 4pt algorithm performs poorly (see Figure 6.8). With 17pts the shape constraints are stronger and features are located more accurately than the CSS method (see Figure 6.9 for example searches).

6.4 Pairwise Reinforcement of Feature Responses (PRFR)

A third approach, known as Pairwise Reinforcement of Feature Responses (PRFR) was developed by the author (see Cristinacce and Cootes [17]). This does not use an explicit shape model, rather it models shape implicitly by learning the pairwise distribution of all true feature locations relative to the best match of each individual feature detector. When searching each true feature location is predicted by multiple detectors. The combination of multiple predictions makes the final prediction of each feature point more robust compared to individual feature search.

6.4.1 Pairwise Feature Distributions

The pairwise distribution $p_{ij}(\mathbf{x}_i|\mathbf{x}_j)$ is defined as the 2D probability density distribution of true feature location i given the best match for feature detector j in the

reference frame defined by the whole face region.

In practice we use histograms of the form $h_{ij}(\mathbf{x}_i - \mathbf{x}_j)$ as an approximation to $p_{ij}(\mathbf{x}_i|\mathbf{x}_j)$. These probability density distribution histograms must be learnt for all possible pairs of feature detector and true feature locations. There are 17 feature detectors, trained to search for 17 feature locations, therefore 289 ($=17 \times 17$) pairwise histograms are required.

Learning of histograms is achieved by applying the global face detector, followed by unconstrained feature detection, to a verification set of face images (see Section 6.1.3 for description of unconstrained search). For each verification image, the true location of all features within the global candidate frame is recorded along with the best match of each feature detector. The ensemble of true feature locations and detector matches allows relative histograms h_{ij} to be computed for the distribution of true feature location i relative to detector j .

Relative histograms h_{ij} for the right eye pupil location, are shown in Figure 6.10. Each diagram plots the distribution of true feature locations relative to the best match of a feature detector (marked with a cross). For example, the spread of true right eye locations relative to a right eye detection are shown in Figure 6.10(a). The spread of right eye locations relative to a left eye detection are shown in Figure 6.10(b). The spread of right eye locations relative to a left eye detection are shown in Figure 6.10(b).

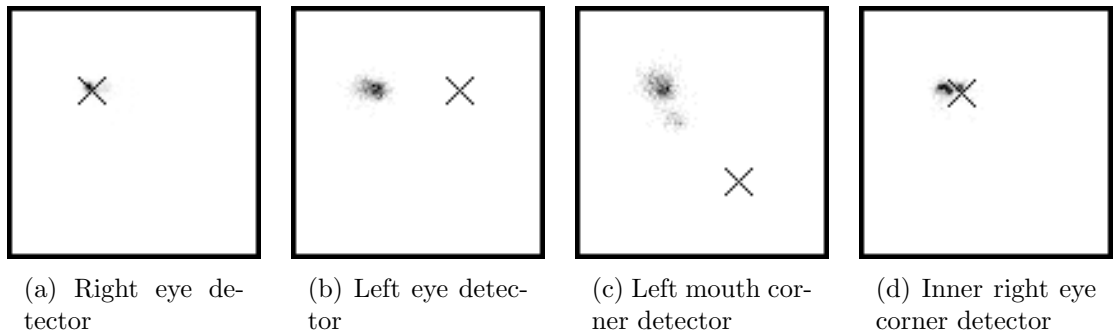


Figure 6.10: Right eye pupil location histograms relative to the best match of four different feature detectors (black pixels indicate peaks in each histogram)

As expected the spread of feature locations for the right eye is least when attempting to predict the right eye location directly with the right eye detector (see Figure 6.10(a)). The histogram of right eye locations relative to predicted location of the left eye (see Figure 6.10(b)) is much less peaked. The spread of each histogram and therefore the confidence in predicting the location of each feature is mainly dependent on three factors, as follows.

1. The variability of the true feature location within the global candidate detector frame.
2. The reliability of the feature detector used to produce the histogram.
3. The distance of the predicted feature point from the feature used to train the detector.

The first factor is a consequence of the natural variation of face shape between individuals and inaccuracies in the window returned by the global face detector. The second factor is a result of noisy feature detectors, which sometimes find false matches. For example the left mouth corner detector (see Figure 6.10(c)) has two peaks for the right eye location, due to tendency of the left mouth corner detector to false match to the left nostril. Similarly the inner right eye corner detector (see Figure 6.10(d)) has multiple peaks due the tendency of this detector to match to the wrong part of the right eye. The third factor occurs because each feature detector is trained locally, so only has limited knowledge of the likely relative location of other facial features.

Using non-parametric histograms allows realistic pairwise statistics to be modelled and makes no prior assumptions as to the distribution of any feature location relative to any particular feature detector. For example Figures 6.10(c) and 6.10(d) show multi-modal histograms which encode variation in the right eye pupil location relative to the more noisy left mouth corner and inner right eye corner feature detectors. This information may have been lost if simpler single Gaussian modelling had been used.

One disadvantage of using histograms is that a reasonably large amount of training data is required to obtain a representative sample of feature location/feature detection pairs. The number of samples required increases with the number of histogram bins. In our experiments, 100x100 bins were used for the whole candidate frame region, trained with 500 verification faces. It may be possible to approximate the distribution histograms using a Gaussian Mixture Model (GMM), if insufficient verification data is available. This would also produce a more compact model. However, in this section we make no Gaussian assumptions.

For reference, Appendix H shows the histograms predicting the locations for the eye pupils, mouth corners and nostrils for the Boosted Cascade Detectors trained on each of the seventeen feature points shown in Figure 6.1.

6.4.2 PRFR Search

Given a set of pairwise histograms h_{ij} , as described in Section 6.4.1, the PRFR algorithm proceeds in a similar manner to the CSS. In that, the global candidate frame predicts local search regions for each detector and each detector j then returns a list of ordered feature detections.

Given an ordered list of detections for each feature detector we wish to predict the location $\hat{\mathbf{x}}_i$ of feature i by combining feature responses with the pairwise probability density distributions $p_{ij}(\mathbf{x}_i|\mathbf{x}_j)$ as follows:-

$$\hat{\mathbf{x}}_i = \arg \max \sum_{j=1}^n \sum_{t=1}^k p_{ij}(\mathbf{x}_i|\mathbf{q}_{jt}) \quad (6.7)$$

Here \mathbf{q}_{jt} is the position of the t^{th} maximum in the response image for feature detector j . We sum the probability density distributions (effectively voting) rather than multiplying, as this generally gives more robust results. Multiplication would be appropriate if all features were independent, which in this case they are not. Note that

the prior probability density distribution $p(\mathbf{q}_{jt})$ of each feature detector is ignored here and only raw matches to the current face region are used to predict the final feature locations ($\hat{\mathbf{x}}_i$).

The first k matches of each feature detector j are used instead of just the best match. This helps to protect against spurious false matches and provides more robust results. By empirical testing a suitable value of k is found to be 3. Similar results are obtained, using any value of k in the range (3, 10). However taking more detections into account increases the time taken to perform PRFR.

In practice the pairwise probability density distributions $p_{ij}(\mathbf{x}_i|\mathbf{x}_j)$ are represented by relative histograms $h_{ij}(\mathbf{x}_i - \mathbf{x}_j)$. When searching, the PRFR algorithm projects the top k feature locations from the j^{th} detector into the histogram frame. Given the feature locations \mathbf{q}_{jt} the relative histogram h_{ij} can be used to predict distributions d_{ijt} of likely locations for feature i . This is summarised by the schematic diagram in Figure 6.11.

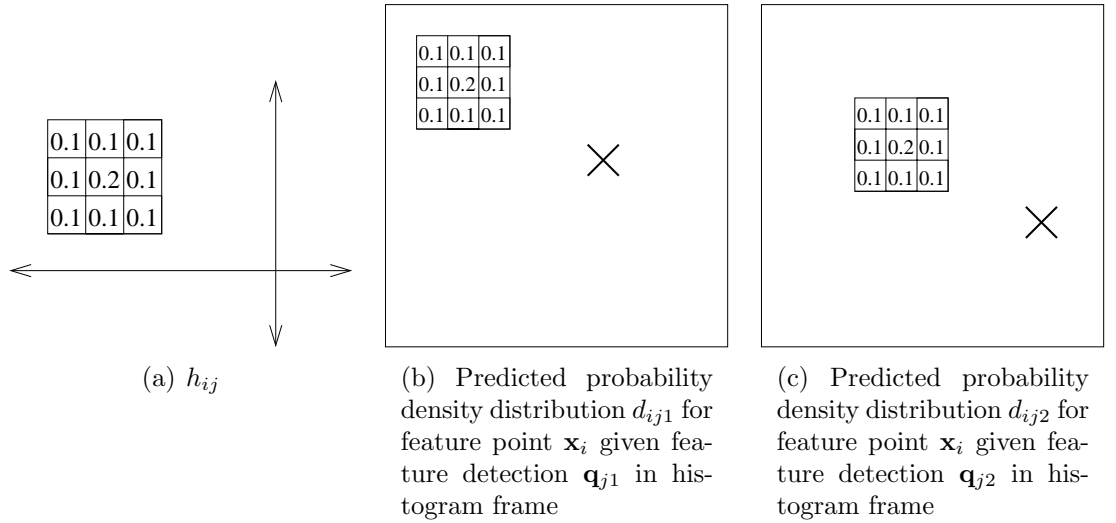


Figure 6.11: Probability density distributions d_{ijt} in the histogram frame for feature point i predicted by relative histogram h_{ij} given the two best matches of detector j

The most likely location $\hat{\mathbf{x}}_i$ is determined by simply summing over all predicted distributions d_{ijt} and selecting the highest ranking pixel in the histogram frame. The

predicted feature locations $\hat{\mathbf{x}}_i$ in the histogram frame can then be mapped back to the corresponding location in the image being searched.

The PRFR method is therefore relatively simple compared to the CSS and SOS algorithms. Both the CSS and SOS require a statistical shape modelling as described in Chapter 5. The PRFR only models shape implicitly by learning the histograms of relative feature locations.

6.4.3 PRFR Results

The results of PRFR are presented in a similar manner to the SOS results of Section 6.3.3. The PRFR method is compared with CSS and accuracy evaluated using the eye pupils and mouth corner features. The mean positional errors m_{e4} for various feature detectors are plotted in Figure 6.12.

Figure 6.12(a) shows that the PRFR 17pt algorithm with Boosted Cascade Detectors is able to predict the location of the eye pupils and mouth corners more accurately than the CSS method with 4 points. However, when the PRFR algorithm uses only four feature detectors, the search is much worse than the CSS method or the average point predictions. Therefore qualitatively the PRFR algorithm with Boosted Cascade Detectors (Figure 6.12(a)) gives similar results to the SOS algorithm (Figure 6.8(a)). Both the PRFR and SOS algorithms perform better than CSS, but only when using all 17 feature detectors.

For example, in Figure 6.12(a) with a threshold of $m_e < 0.15$ the PRFR 17pt method is successful on 96% of faces, compared to 90% with the CSS 4pt method. However, for the PRFR 4pt search only 70% of faces are found successfully, this is much worse performance than the CSS and or even the average points predicted from the global face candidate (83%). Therefore the PRFR needs more than four Boosted Cascade Detectors to be an effective method.

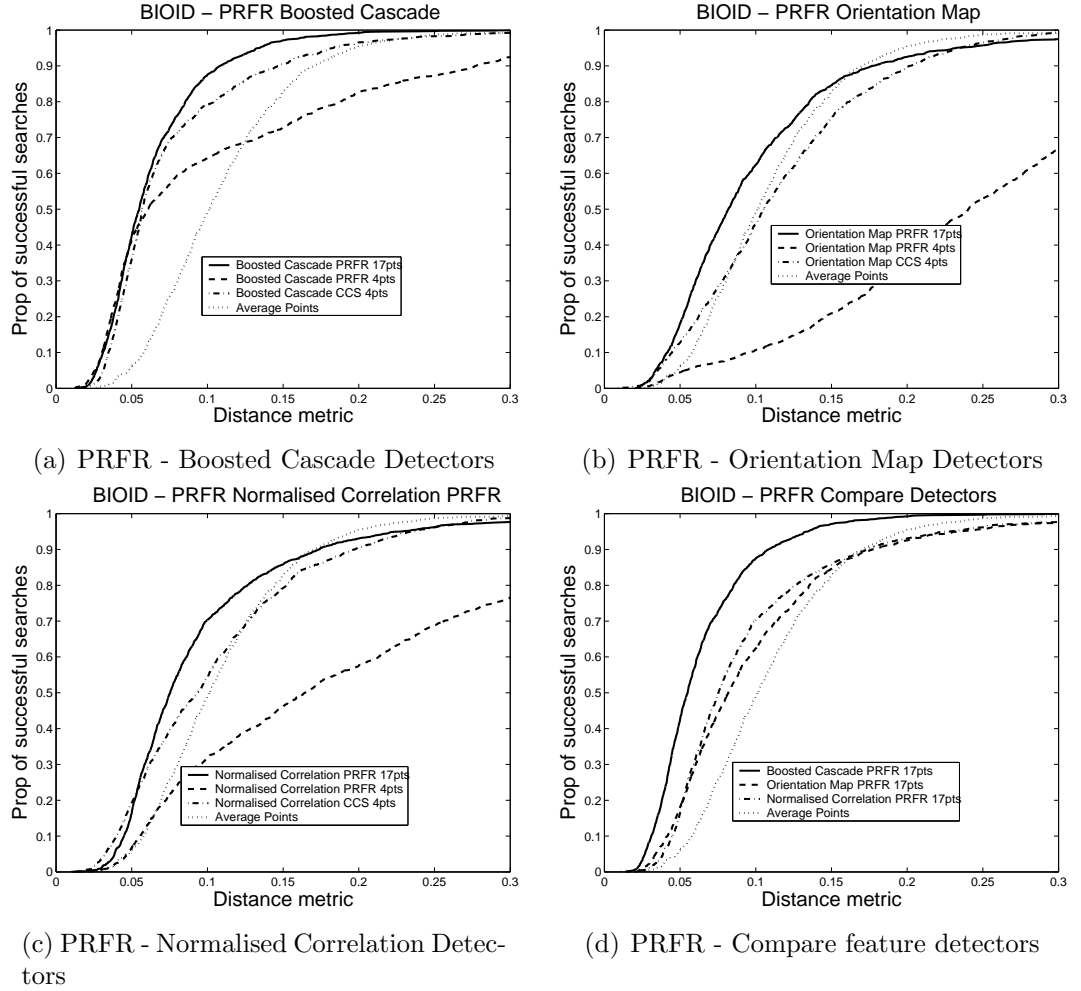


Figure 6.12: Average mean positional error (m_{e4}) when performing PRFR on the BIOID test set

Figures 6.12(b) and 6.12(c) show the performance of the PRFR algorithm with OMDs and NCDs. These graphs show that the PRFR 17pt algorithm still outperforms the CSS 4pt method using both orientation maps and normalised correlation. However the graphs also show that the PRFR 17pt method, does barely better than the average predicted points, using either OMDs or NCDs. The PRFR 4pt algorithm is again much worse than the CSS 4pt method.

For example, with a threshold of $m_e < 0.15$, Figure 6.12(b) shows that the PRFR 17pt OMD method achieves a success rate of only 84%, compared to 83% for the average predicted points. Similarly PRFR 17pt NCD (Figure 6.12(c)), is successful for only 86% of faces at this threshold. However, in both graphs the PRFR 17pt

algorithm gives better performance compared to the CSS 4pt algorithm. There is a large improvement in moving from PRFR 4pt to PRFR 17pt. For Orientation Map Detectors the improvement is 20% \rightarrow 84% (Figure 6.12(c)). For Normalised Correlation Detectors the improvement is 45% \rightarrow 86% (Figure 6.12(c)).

Figure 6.12(d) confirms that PRFR 17pt with BCDs is by far the most effective method. Finding features accurately for 96% of faces, compared to 86% with NCDs and 84% with OMDs. Therefore the PRFR 17pt method works well, only if many feature detectors are used and the detectors are reasonably accurate. The only feature detector found to be reliable enough in Figure 6.12 is the BCD.

6.4.4 PRFR Search Examples

Examples of the PRFR algorithm improving on the results of the CSS method are shown in Figure 6.13.

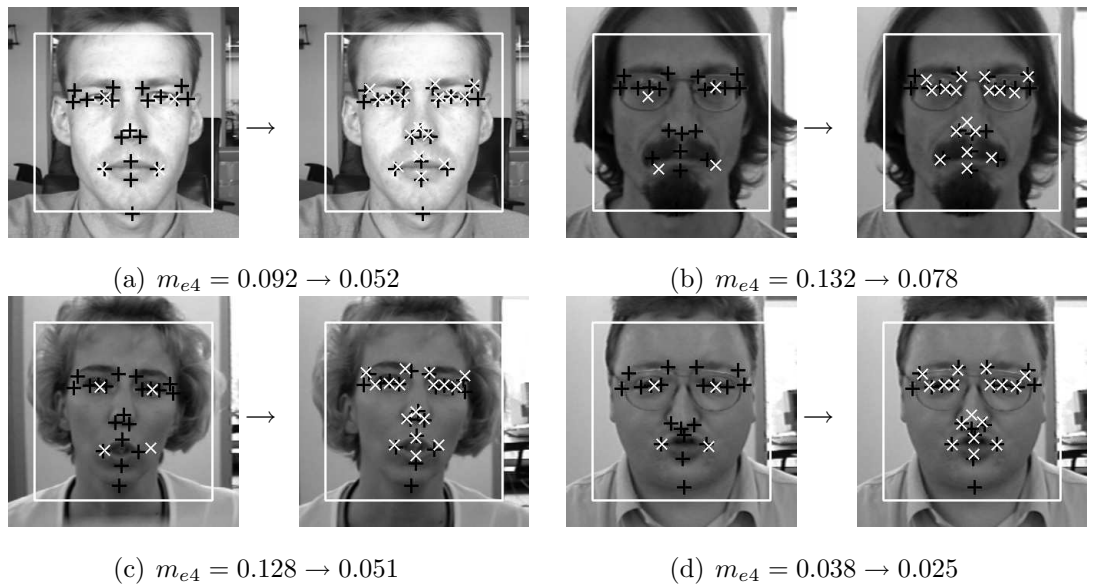


Figure 6.13: Examples of improvement in search accuracy m_{e4} using PRFR 17pt search, compared to CSS 4pt search with Boosted Cascade Detectors.

Figure 6.13 compares PRFR 17pt BCD with CSS search on the same images as

Figure 6.9, which compares SOS with CSS. Like the SOS 17pt method, the PRFR 17pt method improves on the CSS search accuracy (m_{e4}). For example, in Figure 6.13(a) the eyes are found more accurately by PRFR compared to CSS. In Figures 6.13(b) and 6.13(c) the mouth locations are improved using PRFR compared to CSS. There is also a small improvement in Figure 6.13(d).

6.4.5 PRFR Timing

The speed of the PRFR algorithm using Boosted Cascade Detectors is evaluated on a PII 500Mhz PC. The results are summarised in Table 6.4.

Event	PRFR 4pt	PRFR 17pt
Global search	~400ms	~400ms
Local feature search	~50ms	~150ms
Histogram Voting	~150ms	~700ms
Total	~600ms	~1250ms

Table 6.4: Time to perform PRFR with BCDs on a single BIOID image using a 500Mhz PII processor

Table 6.4 shows that the global detector and PRFR 17pt local search find features in ~1250ms. With 4pts the PRFR predicts features in ~600ms, but performs very badly. The most time consuming step is summing the histograms (~700ms). This is mainly due to the large number and size of each histogram image. For the PRFR 17pt method there are 2890 (=17x17x10) histogram additions per search. However, the PRFR 17pt method in its current form, would still be able to search multiple images in less than a second on a modern PC.

6.4.6 PRFR Conclusions

The PRFR 17pt algorithm is shown to provide superior performance to the CSS 4pt algorithm using BCDs. However, when using weaker feature detectors, such as the

OMD and NCD the performance of the PRFR 17pt algorithm is relatively poor (see Figure 6.12). Therefore the PRFR algorithm only works with 17 feature points and a relatively strong feature detector, such as the Boosted Cascade Detector.

Recently Chen *et al.* [5] have developed a similar approach to the PRFR method. Chen *et al.* use local BCD search, but do not use shape constraints. Instead boosting chain learning [98] is used to calculate a probabilistic like output for each detector. Chen *et al.* report good results on the FERET data set [67]. However, this test set contains relatively clean image set, so it is unclear how well the method compares to the PRFR method described here.

6.5 Comparison of SOS and PRFR Algorithms

Sections 6.3.3 and 6.4.3 show that the SOS 17pt and PRFR 17pt algorithms outperform the CSS 4pt method. The metric used (m_{e4}) is the point to point error of the eye pupils and mouth corners. However, the SOS 17pt and PRFR 17pt methods are able to predict all 17 feature points shown in Figure 6.1, not just the eye pupils and mouth corners. Therefore it makes sense to compare the SOS and PRFR algorithms by computing the point to point error m_{e17} over all 17 feature points.

6.5.1 SOS and PRFR 17pt Results

The results comparing PRFR and SOS with the 17pt metric (m_{e17}) are shown in Figure 6.14.

Figure 6.14(a) compares the SOS and PRFR algorithms with the strongest feature detectors, i.e. Boosted Cascade Detectors. The graph shows that PRFR+BCDs gives better performance than SOS+BCDs. For example at an accuracy threshold of $m_{e17} < 0.15$ the success rate for PRFR is 98% compared to 96% for SOS. The PRFR method has a greater success rate at all values of m_{e17} . Both the PRFR and SOS

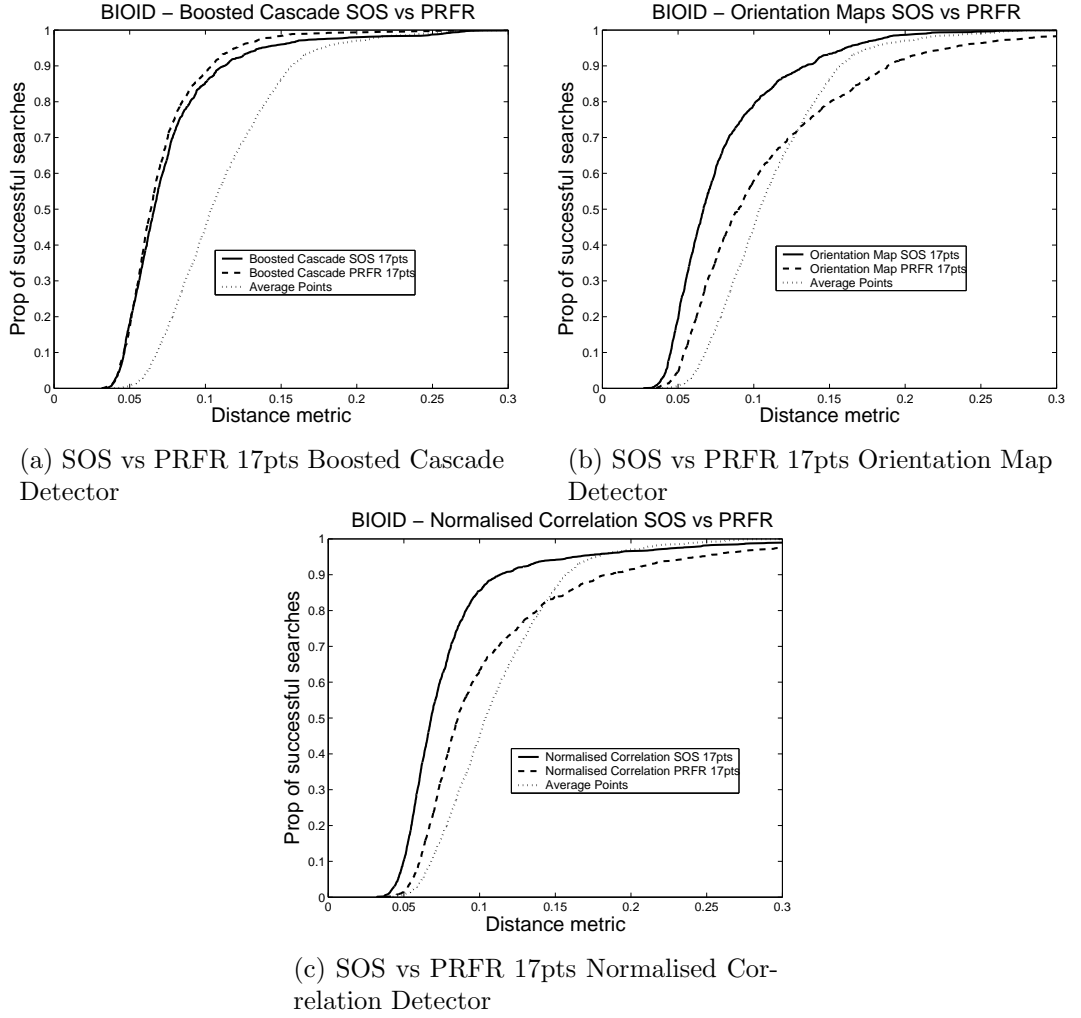


Figure 6.14: Comparison of PRFR and SOS algorithms on the BIOID test set given the same feature detector, using the average mean positional error of all 17pts (m_{e17})

methods are much more accurate than the average points predicted by the global candidate (see Figure 6.14(a)).

However Figures 6.14(b) and 6.14(c) show a different story. With weaker feature detectors, such as Orientation Map Detectors and Normalised Correlation Detectors the PRFR algorithm performs very poorly and the SOS algorithm gives much better results. For example with orientation maps (Figure 6.14(b)) and $m_{e17} < 0.15$, the search is successful in 93% of cases using SOS, 85% using average point prediction, but only 80% using PRFR. With normalised correlation (Figure 6.14(c)) 94% of cases succeed using SOS, but only 82% using PRFR.

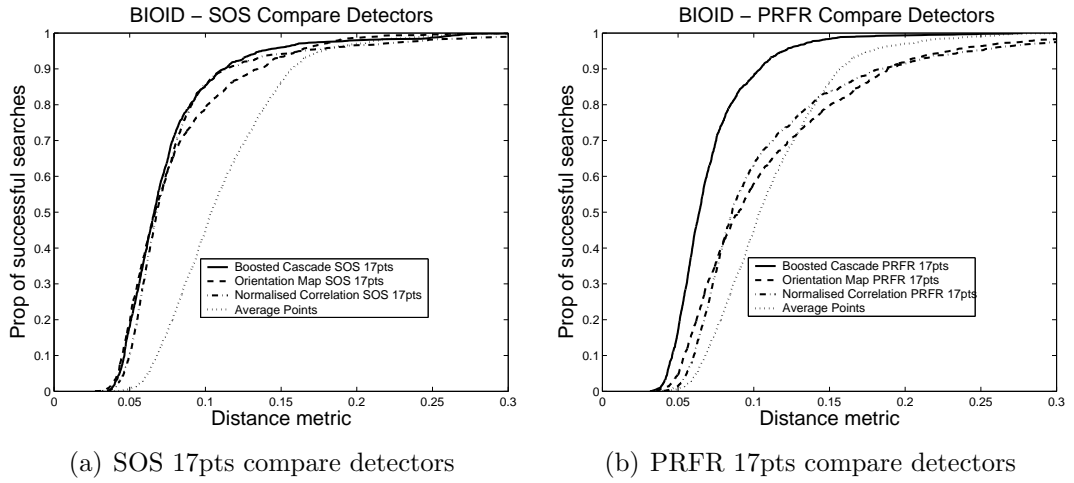


Figure 6.15: Comparing performance of PRFR and SOS algorithms with different feature detectors on the BIOID test set, using the average mean positional error of all 17pts (m_{e17})

Therefore SOS is far more reliable than PRFR. Given weak detectors, such as orientation maps and normalised correlation, the SOS search still performs well. However with stronger feature detectors, such as Boosted Cascade Detectors the PRFR is slightly more accurate than the SOS algorithm. The reliability of the SOS algorithm with any feature detector is made clear in Figure 6.15(a), which shows the SOS always giving better results compared to the average points. Figure 6.15(b) shows that only PRFR combined with Boosted Cascade Detectors gives better than average feature detection.

6.5.2 SOS and PRFR Search Examples

Figure 6.16 illustrates the different results obtained when predicting feature locations with SOS and PRFR using different feature detectors.

Figures 6.16(a) , 6.16(b) and 6.16(c) show SOS gives very similar accuracy with all three feature detectors. Figure 6.16(d) shows PRFR gives accurate point prediction with Boosted Cascade Detectors. However, as indicated by the graphs in Figure 6.15(b), the search accuracy with Orientation Map Detectors and Normalised Correlation Detectors is degraded when using PRFR (see Figures 6.16(e) and Fig-

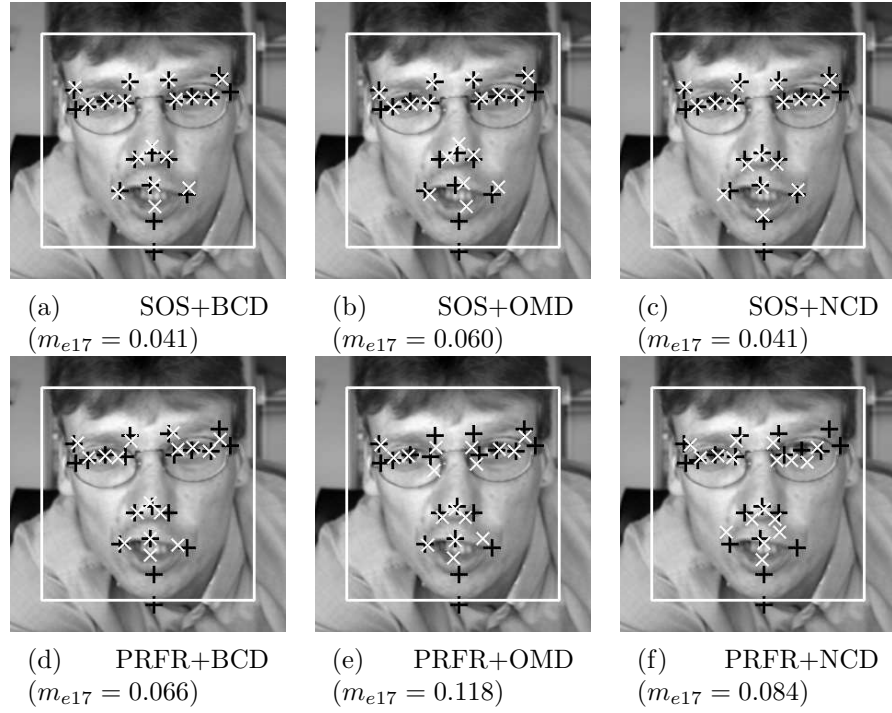


Figure 6.16: Examples of SOS and PRFR 17pt search with different feature detectors

ures 6.16(f)).

6.5.3 Individual Feature Errors

The graphs in Figure 6.14 compare the average errors m_{e17} of the SOS and PRFR algorithms over all 17 features. It is also possible to examine the individual point to point errors for individual feature points. These errors are plotted in Figure 6.17.

Figure 6.17 shows that for each feature PRFR and SOS search always perform much better than the average point predictor. The PRFR tends to be slightly better than the SOS for most features. Hence the average error m_{e17} over all images and all features being 7.07% for PRFR, 7.55% for SOS and 11.08% using average point prediction. Figure 6.17 also shows that the errors for all three point prediction methods are greater for the nose and mouth features and less for the eye and eye brow regions.

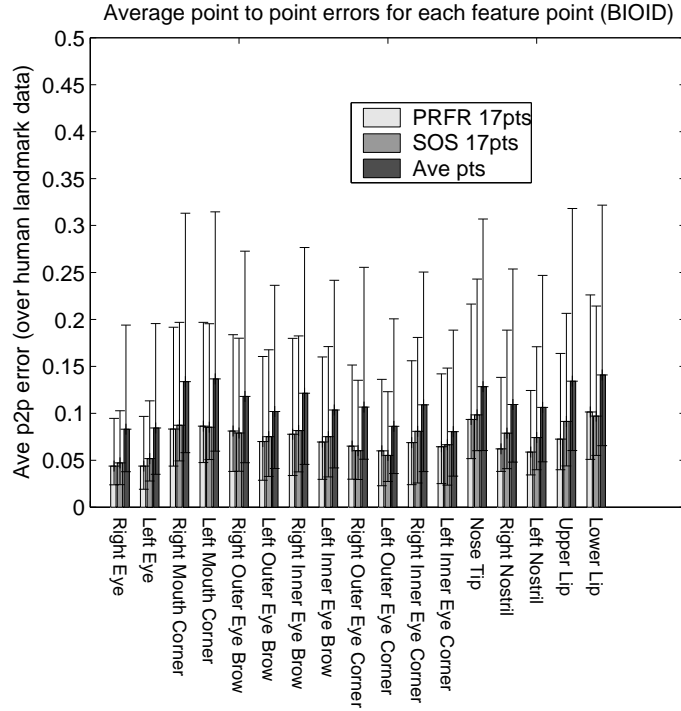


Figure 6.17: Average point to point errors m_{e1} for each feature when performing PRFR, SOS and average point prediction on the BIOID test set (error bars show upper and lower quartiles of m_{e1} for each feature).

6.6 Conclusions

This chapter demonstrates that shape constraints are able to improve the accuracy of local feature detectors. Unconstrained local feature detectors perform poorly giving less accurate results than the average points predicted from the whole face template.

Three methods of combining local feature detectors with shape constraints are described, namely Combinatoric Shape Search (CSS), Shape Optimised Search (SOS) and Pairwise Reinforcement of Feature Responses (PRFR). These three techniques can be combined with any of three different types of local feature detector, namely Boosted Cascade Detectors (BCDs), Orientation Map Detectors (OMDs) or Normalised Correlation Detectors (NCDs). The most accurate combination is PRFR+BCDs. However the SOS algorithm is shown to perform much better than PRFR with weak feature detectors such as the OMDs or NCDs.

The accuracy of PRFR with BCDs is probably due to the fact that the histogram shape constraints are relatively loose. Therefore the PRFR is able to match more closely to a large variety of faces compared to the SOS. The SOS shape constraints are stronger, which could explain the superior performance with weaker detectors such as the OMDs and NCDs.

With BCDs the PRFR is only slightly more accurate than the SOS, however the SOS is faster ($\sim 550\text{-}750\text{ms}$ vs $\sim 1250\text{ms}$). Therefore when deciding between the two methods, there is a compromise between performance and speed.

Figure 6.17 shows that the features that are most difficult to locate using either PRFR or SOS are the mouth and nose features, in particular the nose tip and upper/lower lip. If the overall average search error m_{e17} is to be improved these features need to be located more accurately.

Chapter 7

Active Appearance Models

The previous chapter described the use of shape modelling to constrain feature detection. However, another well established method for combining shape and texture information is the Active Appearance Model (AAM) [6] search method. An overview of this technique is given in Section 2.6 of the literature review in Chapter 2. In this chapter the AAM is compared with the feature detection methods described in Chapter 6.

7.1 Appearance Models (APMs)

Appearance Models (APMs) are a combined model of both the shape and texture of the face. Separate shape and texture models are built using Principal Components Analysis (PCA) and then combined using a further PCA.

7.1.1 Shape Model

The shape model is built using the method described in Chapter 5 using the WEB-CAM training set. This produces a linear model of shape variation, where a shape

example \mathbf{x} is represented by a shape parameter vector \mathbf{b}_s , as follows:-

$$\mathbf{x} = \bar{\mathbf{x}} + \mathbf{P}_s \mathbf{b}_s \quad (7.1)$$

Where $\bar{\mathbf{x}}$ is the mean shape, \mathbf{P}_s is a set of orthogonal modes of variation and \mathbf{b}_s is a set of shape parameters. The first two modes of variation of the shape model, trained on the WEBCAM data, are shown in Figure 7.1.

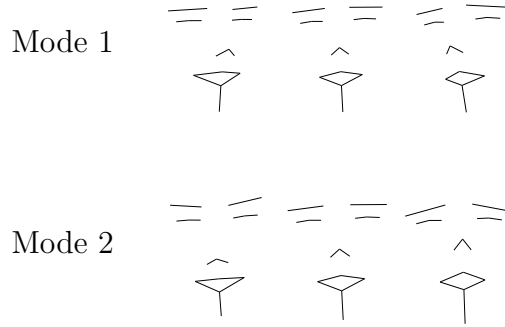


Figure 7.1: Modes of Shape Model ($\pm 2.5\text{std}$)

7.1.2 Texture Model

The texture model is built by warping each training example into the mean shape, to produce a “shape free” face patch. The texture can then be sampled to form a vector of grey values \mathbf{g} . This texture vector is normalised to minimise the effect of global lighting variation, by applying a scaling α and an offset β .

$$\mathbf{g} = (\mathbf{g}_{im} - \beta \mathbf{1}) / \alpha \quad (7.2)$$

The values of α and β are computed as follows.

$$\alpha = \mathbf{g}_{im} \cdot \bar{\mathbf{g}} \quad , \quad \beta = (\mathbf{g}_{im} \cdot \mathbf{1})/n \quad (7.3)$$

PCA is then applied to the normalised texture to produce a linear model of the shape-free texture variation (see Equation 7.4).

$$\mathbf{g} = \bar{\mathbf{g}} + \mathbf{P}_g \mathbf{b}_g \quad (7.4)$$

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

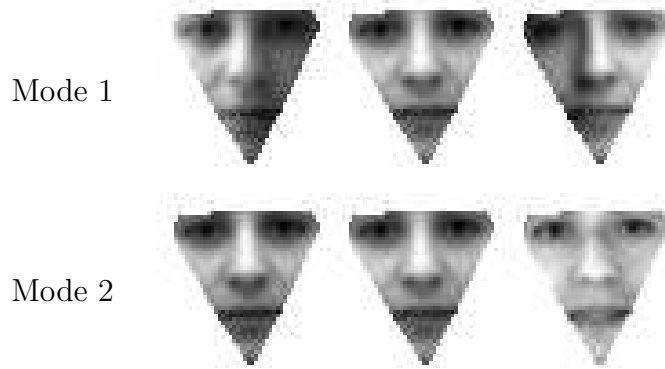


Figure 7.2: Modes of Texture Model ($\pm 2.5\text{std}$)

7.1.3 Combined Model

Following Edwards *et al.* [21], the combined model is formed by applying a further PCA to find correlations between the shape model and texture model parameters. The first step is to concatenate the shape parameters \mathbf{b}_s and texture parameters \mathbf{b}_g to form a combined vector \mathbf{b} as follows.

$$\mathbf{b} = \begin{pmatrix} \mathbf{W}_s \mathbf{b}_s \\ \mathbf{b}_g \end{pmatrix} = \begin{pmatrix} \mathbf{W}_s \mathbf{P}_s^T (\mathbf{x} - \bar{\mathbf{x}}) \\ \mathbf{P}_g^T (\mathbf{g} - \bar{\mathbf{g}}) \end{pmatrix} \quad (7.5)$$

Here $\mathbf{W}_s = r\mathbf{I}$ is a diagonal weight matrix allowing for the difference in units between shape and texture parameters, where r^2 is the ratio of the total intensity variation to the total shape variation (in the normalised frames).

A further PCA is then performed on the set of combined vectors \mathbf{b} . To produce a linear model of the variation in \mathbf{b} , as follows.

$$\mathbf{b} = \mathbf{P}_c \mathbf{c} \quad (7.6)$$

where \mathbf{P}_c are the eigenvectors and \mathbf{c} is a vector of appearance parameters controlling both the shape and grey-levels of the model. Since the shape and grey-model parameters have zero mean, \mathbf{c} is also zero mean.

The shape \mathbf{x} and \mathbf{g} texture vectors can now be expressed as functions of \mathbf{c} , as shown in Equation 7.7.

$$\begin{aligned} \mathbf{x} &= \bar{\mathbf{x}} + \mathbf{P}_s \mathbf{W}_s^{-1} \mathbf{P}_{cs} \mathbf{c} \\ \mathbf{g} &= \bar{\mathbf{g}} + \mathbf{P}_g \mathbf{P}_{cg} \mathbf{c} \end{aligned} \quad (7.7)$$

where

$$\mathbf{P}_c = \begin{pmatrix} \mathbf{P}_{cs} \\ \mathbf{P}_{cg} \end{pmatrix} \quad (7.8)$$

By varying the individual parameters of \mathbf{c} the principal modes of variation are constructed. The first two modes are shown below.

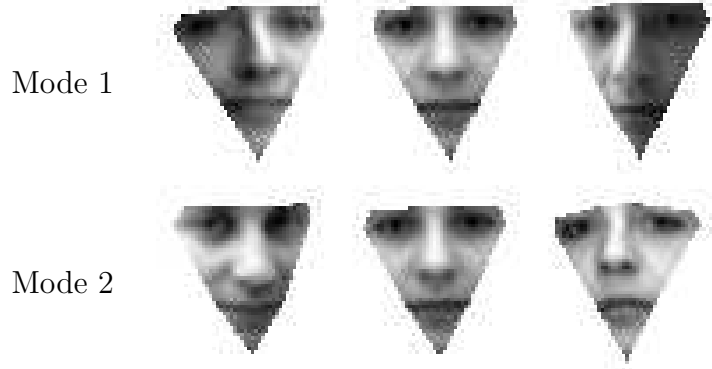


Figure 7.3: Modes of Combined Model ($\pm 2.5\text{std}$)

7.2 Active Appearance Models (AAMs)

The Appearance Models (APMs) model variation given a marked up training set of faces. However given an unlabelled face the AAM search is required to match the APM to the face image automatically.

7.2.1 AAM Search

The AAM is *active* because the parameters \mathbf{c} of the APM (see Section 7.1) are altered such that the appearance model moves and evolves in the image plane, hopefully converging to produce the best possible match of the APM to the face image. The update rule for the \mathbf{c} parameters is based on the texture residual between the model texture and image pixels.

More specifically, given a set of appearance parameters \mathbf{c} , the current APM face can be reconstructed using Equation 7.7. This face is then projected into the image being searched. The idea of the AAM algorithm is to iteratively compare the texture sampled from the image \mathbf{g}_{im} to the current model texture \mathbf{g}_{apm} and update the APM parameters. If the correct direction in parameter space is chosen the APM will converge to the best possible solution. Formally, at each iteration the texture residual $\mathbf{r}(\mathbf{p})$ is computed, as follows.

$$\mathbf{r}(\mathbf{p}) = \mathbf{g}_{im} - \mathbf{g}_{apm} \quad (7.9)$$

Where \mathbf{p} are the current parameters of the model, $\mathbf{p}^T = (\mathbf{c}^T | \mathbf{t}^T | \mathbf{u}^T)$. Where \mathbf{c} are the PCA parameters of the statistical model, \mathbf{t} describes the project from the model plane into the image plane and \mathbf{u} describes the texture normalisation parameters α and β (see [10] for more details). Having computed $\mathbf{r}(\mathbf{p})$ the AAM search predicts the direction of movement in parameter space, using an update matrix \mathbf{R} .

$$\delta \mathbf{p} = \mathbf{R} \mathbf{r}(\mathbf{p}) \quad (7.10)$$

The parameter vector \mathbf{p} is then updated and the APM re-projected into the image. The search terminates when the change in parameters $\delta \mathbf{p}$ drops below a threshold.

7.2.2 AAM Training

Training the AAM involves computing the update matrix \mathbf{R} described in 7.10. The matrix \mathbf{R} models the relationship between the texture residual $\mathbf{r}(\mathbf{p})$ and the change $\delta \mathbf{p}$ required to improve the sum of squares magnitude of the residuals. A Taylor's series expansion of $\mathbf{r}(\mathbf{p})$ gives

$$\mathbf{r}(\mathbf{p} + \delta \mathbf{p}) = \mathbf{r}(\mathbf{p}) + \frac{\partial \mathbf{r}}{\partial \mathbf{p}} \delta \mathbf{p} \quad (7.11)$$

Where the ij^{th} element of matrix $\frac{\partial \mathbf{r}}{\partial \mathbf{p}}$ is $\frac{dr_i}{dp_j}$. Setting the right hand side of Equation 7.11 to zero and computing the pseudo inverse of $\frac{\partial \mathbf{r}}{\partial \mathbf{p}}$ leads to the following solution for $\delta \mathbf{p}$.

$$\delta \mathbf{p} = -\mathbf{R} \mathbf{r}(\mathbf{p}) \quad \text{where} \quad \mathbf{R} = \left(\frac{\partial \mathbf{r}}{\partial \mathbf{p}}^T \frac{\partial \mathbf{r}}{\partial \mathbf{p}} \right)^{-1} \frac{\partial \mathbf{r}}{\partial \mathbf{p}}^T \quad (7.12)$$

Since \mathbf{R} is computed in a standard reference frame it is assumed to be constant throughout an incremental search. Therefore it is pre-computed from the training set off-line. This is achieved by fitting the appearance model to each training example to obtain vector \mathbf{p} and systematically varying each appearance parameter p_i to obtain a numeric approximation to $\frac{\partial \mathbf{r}}{\partial \mathbf{p}}$.

7.3 AAM Results

The AAM search is evaluated in a similar manner to the CSS, SOS and PRFR feature detector based algorithms described in Chapter 6. The Boosted Cascade Face Detector is used to predict the approximate face location and the AAM is initialised by fitting to the average points within the face template region. The accuracy of AAM search is then evaluated using the mean point to point distance error m_{e17} described in Section 6.1.4. There are many different formulations of the AAM. In this chapter, the effects of varying the resolution of the APM, the face region modelled and the texture sampling are evaluated.

7.3.1 Vary Model Resolution

When computing the APM, the face region lying within the set of landmark points is subsampled to build the grey scale models described in Section 7.1.2. The number of pixels sampled from each face can be varied, to build higher or lower resolution models. The average faces of the APM models with different resolutions are shown in Figure 7.4.

Figure 7.4(c) shows the increase in detail in the average model face when sampling 5000 pixels compared sampling only 1000 pixels (Figure 7.4(a)). The improvement in performance when using a greater resolution model is shown in Figure 7.5.

Figure 7.5 shows that higher resolution models tend to perform better than lower

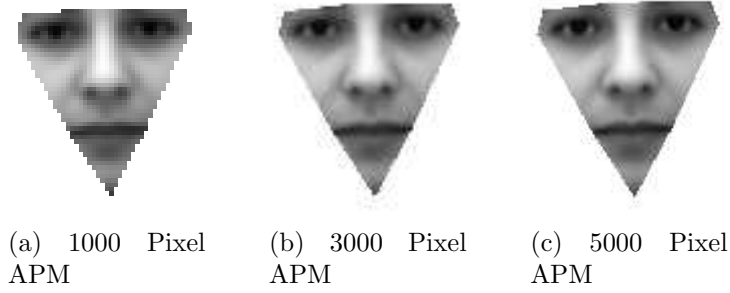


Figure 7.4: Mean face of APMs, built at different resolutions

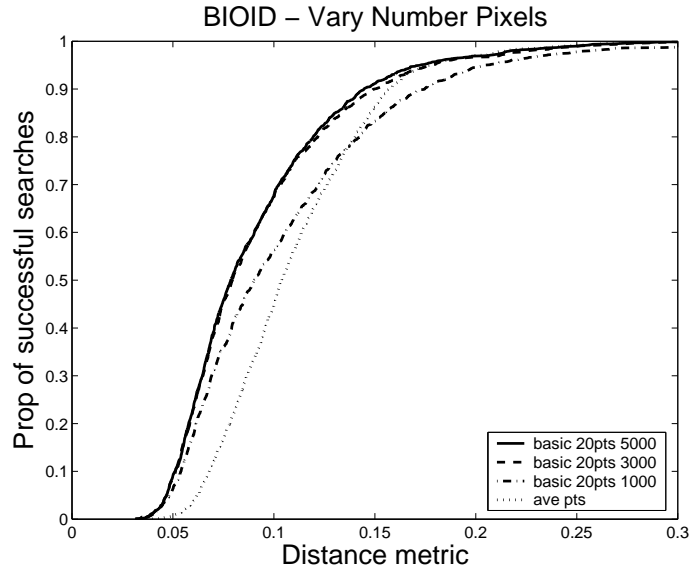


Figure 7.5: Point to point error m_{e17} when performing AAM search on the BIOID test set, varying the resolution of the APM

resolutions. For example with a proximity threshold $m_{e17} < 0.15$ the AAM search with 1000 pixels is successful in only 82% of cases. This is actually worse than the average points predicted from the global face region, without any local search which has a success rate of 85%. When increasing the resolution from 1000 \rightarrow 3000 pixels the success rate increases from 82% \rightarrow 90%. When the resolution is further increased 3000 \rightarrow 5000 pixels there is a much smaller improvement to the success rate 90% \rightarrow 91%. Therefore Figure 7.5 indicates that search is much improved when sampling 3000 pixels instead of 1000, but the basic AAM performance cannot be significantly improved by increasing the resolution further.

7.3.2 Vary Region Modelled

Another parameter of the AAM is the region modelled by the appearance model. In Section 7.3.1, the APM is built using the 20pt markup scheme, see Figure 7.6(a). It is also possible to increase the size of the region slightly by moving the temple points to the hairline and adding two new points at the corners of the jaw. This alternative markup scheme is shown in Figure 7.6(b).

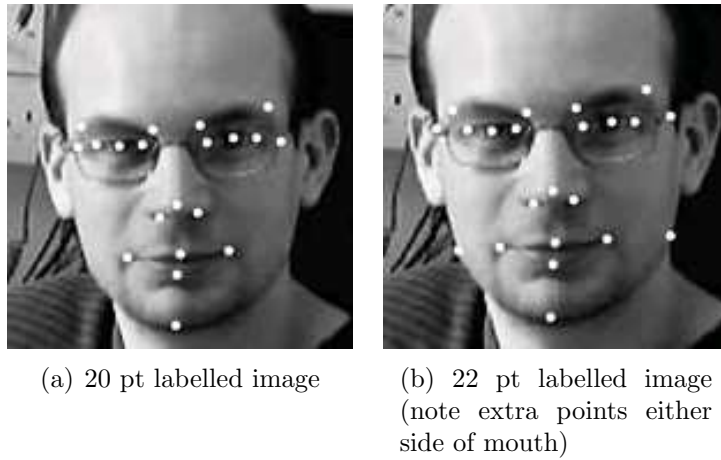


Figure 7.6: Two different labelling schemes

The markup scheme in Figure 7.6(b) increases the proportion of the face modelled by the APM. The new modes of variation using this region are shown in Figure 7.7 and can be compared with the original modes of variation shown in Figure 7.3.

The effect of modelling a larger face region on the search accuracy is shown in Figure 7.8. Here the same sampling rate of 1000 pixels is used for both APMs.

Figure 7.8 shows that the basic AAM search with 1000 pixels is much improved when using the new 22 point markup scheme. Using a proximity threshold of $m_{e17} < 0.15$, the success rate is 92% using the 22 point region compared to 82% using the 20 point region. The 22 point gives similar performance to the higher resolution 20 point models shown in Figure 7.3.1. Therefore Figure 7.8 indicates that modelling a larger face region can improve AAM search.

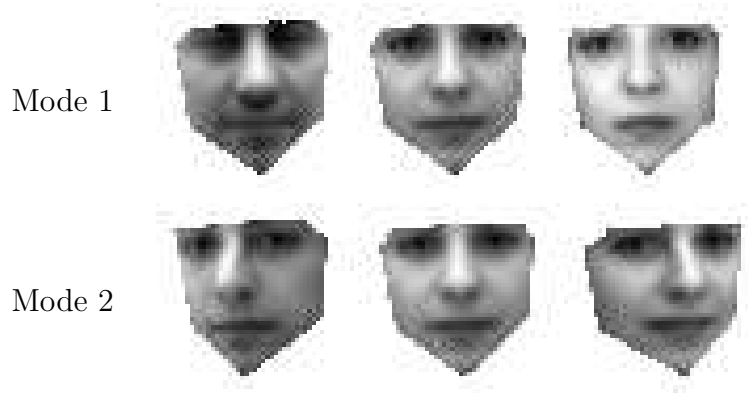


Figure 7.7: Modes of Combined Model 22pts, sampling 1000 pixels ($\pm 2.5\text{std}$)

7.3.3 Vary Sampling Method

Another variation of the APM/AAM method, is to vary the way texture is sampled within the face region. The basic AAM uses a linear normalisation of the sampled texture vector, in attempt to minimise the effect of global lighting variation (see Section 7.1.2). However it is possible to employ more sophisticated texture processing.

For example, Cootes and Taylor [12] describe a method using local image gradients instead of raw pixel values to drive the AAM search. Scott *et al.* [82] go further and compute local gradients and also edge/corner features based around the Harris corner detector [40]. In this section, the improvement in performance when using the gradient/corner/edge sampling method due to Scott *et al.* [82] is compared with pixel based AAM sampling. The method computes four values for each pixel namely, g'_x the normalised gradient in the x direction, g'_y the normalised gradient in the y direction, e' a measure of “edgeness” and c' a measure of “corneriness”.

The raw gradients g_x and g_y are computed from the texture vector and the gradients are normalised as follows.

$$g'_x = \frac{g_x}{|g| + |\bar{g}|} \quad g'_y = \frac{g_y}{|g| + |\bar{g}|} \quad (7.13)$$

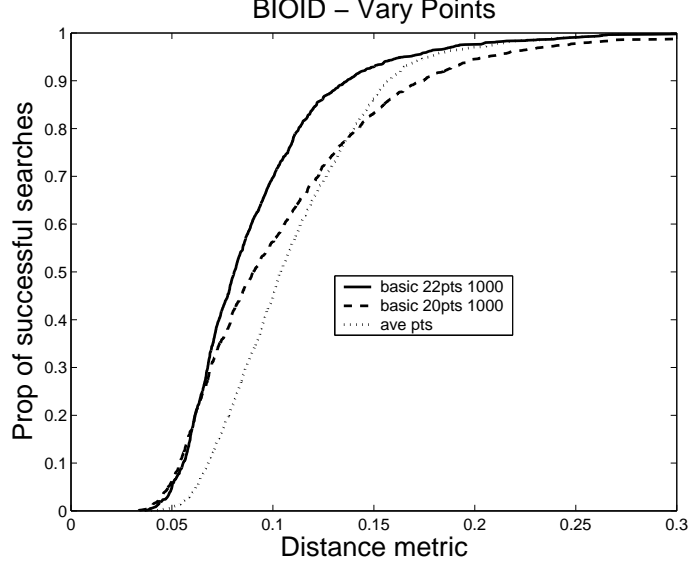


Figure 7.8: Point to point error m_{e17} when performing AAM search on the BIOID test set, varying the face region modelled by the APM

Where $\mathbf{g} = (g_x, g_y)^T$ at each pixel and $|\bar{\mathbf{g}}|$ is computed over the entire model region. The corner and edge values are based on the Harris corner detector [40], which finds the eigenvalues of the local gradient matrix.

$$M = \begin{pmatrix} A & C \\ C & B \end{pmatrix} = \begin{pmatrix} g_x^2 & g_x g_y \\ g_x g_y & g_y^2 \end{pmatrix} \quad (7.14)$$

The edginess, e , and cornerness, c , values are computed from this matrix as shown in Equation 7.15. Additionally the edge and corner measures are normalised as shown in Equation 7.16. For further details of the derivation of these equations see Scott *et al.* [82].

$$e = 2AB - 2C^2 \quad c = (A + B)\sqrt{(A - B)^2 + 4C^2} \quad (7.15)$$

$$e' = \frac{e}{e+\bar{e}} \quad c' = \frac{c}{c+\bar{c}} \quad (7.16)$$

The AAM algorithm then proceeds as before. However instead of using the normalised pixel texture vector to drive the AAM, a new APM is constructed using the gradient/edge/corner values for each pixel and this new texture vector used to drive the AAM search. The aim is to use edge and corner information more explicitly as distinct edges and corners are present on the human face. The effect of edge/corner texture sampling is shown in Figure 7.9.

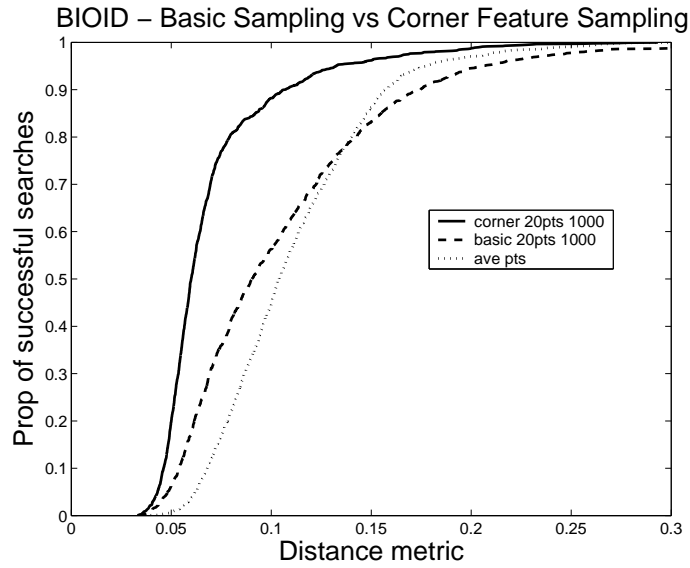


Figure 7.9: Point to point error m_{e17} when performing AAM search on the BIOID test set, varying the texture sampling method

Figure 7.9 shows that there is a large increase in search accuracy when using edge/corner sampling compared to the basic AAM search, when using the 20 point label scheme and modelling 1000 pixels. With edge/corner sampling and a proximity threshold of $m_{e17} < 0.15$ the success rate is 96%, which compares to just 83% using normal sampling over the same facial region. Figure 7.9 shows that edge/corner sampling method is more successful at all m_{e17} thresholds. Therefore edge/corner sampling is superior to normal texture sampling.

7.3.4 Optimal AAM Formulation

Section 7.3.3 shows that moving to edge/corner sampling gives a clear improvement over normal AAM texture sampling. Therefore this section attempts to optimise AAM performance using edge/corner sampling by varying the other search parameters. For example Figure 7.10(a) compares the performance of the 20 point model and the 22 point model regions.

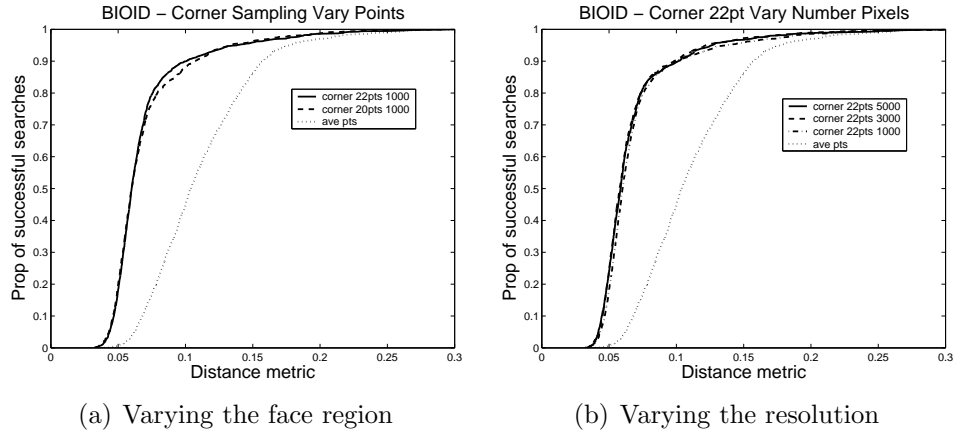


Figure 7.10: Point to point error m_{e17} when performing AAM search on the BIOID test set using corner texture sampling

Figure 7.10(a) shows that when using the edge/corner sampling, moving from the 20 point markup scheme to the 22 point markup scheme makes little difference. There was a large increase when using the 22 point region instead of the 20 point region with normal texture sampling (see Figure 7.8). The reason for the lack of improvement is probably due to the fact that the extra facial region modelled with 22 points is the cheek region, which does not contain many strong edges or corners. Also the edge/corner sampling AAM with 20 point markup performs strongly, so it is more difficult to improve the search.

Figure 7.10(b) shows using higher resolution models with edge/corner sampling makes a small improvement when moving from 1000 \rightarrow 3000 pixels. The success rate increase marginally from 96% \rightarrow 97%, using a proximity threshold of $m_{e17} < 0.15$ (see

Figure 7.10(b)). However virtually identical search accuracy is shown when using 3000 or 5000 pixels. This is similar to the result shown by Figure 7.5 and indicates that the AAM search is not improved by modelling more than 3000 pixels.

7.4 AAM Timings

The search time using the various formulations of the APM/AAM search are shown in Table 7.1, which shows that the time to perform the local AAM search is primarily dependent upon the resolution of the APM and the sampling method.

Event	Time
Global search	~400ms
Local Basic 20pt 1000 pixel search	~80-100ms
Local Basic 20pt 3000 pixel search	~150-200ms
Local Basic 20pt 5000 pixel search	~240-280ms
Local Edge/Corner 22pt 1000 pixel search	~390-450ms
Local Edge/Corner 22pt 3000 pixel search	~800-1400ms
Local Edge/Corner 22pt 5000 pixel search	~1800-3000ms

Table 7.1: Time to perform various AAM searches on a single BIOID image using a 500Mhz PII processor

For example, the quickest search method is basic texture sampling of 1000 pixels (using the 20pt labelling scheme). When basic sampling is increased from 1000 to 5000 pixels the search time is approximately 2.5 times larger. Using edge/corner sampling is much slower than basic sampling, because 4 values have to be computed for every pixel. Therefore edge/corner sampling is much slower than basic sampling - ~390-450ms vs ~80-100ms at a resolution of 1000 pixels. With edge/corner sampling the search time becomes prohibitive when increasing the resolution beyond 1000 pixels (see Table 7.1).

7.5 AAM Conclusions

Figure 7.9 shows that edge/corner sampling gives much improved search results compared to basic texture sampling. Figure 7.10 shows that increasing the model resolution or changing to the 22 point markup scheme makes little difference to the search accuracy of the edge/corner AAM.

Table 7.1 shows that edge/corner sampling is more computationally expensive compared to basic texture sampling. Also, the search time is vastly increased when modelling more than 1000 pixels with edge/corner AAMs.

However, given the improvement in search accuracy using edge/corner sampling, a sensible compromise between speed and accuracy is to use edge/corner sampling, with the 22pt markup, but limit the resolution of the face region to 1000 pixels. An AAM using this formulation is able to find feature points accurately and the total search time (global + local = $\sim 790\text{-}850\text{ms}$) is less than a second using a PII 500Mhz processor.

7.6 Combining Feature Detection and the AAM

This section compares the AAM search with the feature detector based methods described in Chapter 6 and also present results when the two approaches are combined.

7.6.1 Comparing AAM, SOS and PRFR

Figure 7.11 compares the accuracy of the edge/corner AAM and basic AAM search (using 1000 pixels and 22pt markup) with the SOS and PRFR constrained feature detection methods, using Boosted Cascade Detectors (see Sections 6.3 and 6.4).

Figure 7.11(a) shows that the SOS, PRFR methods perform better than the basic

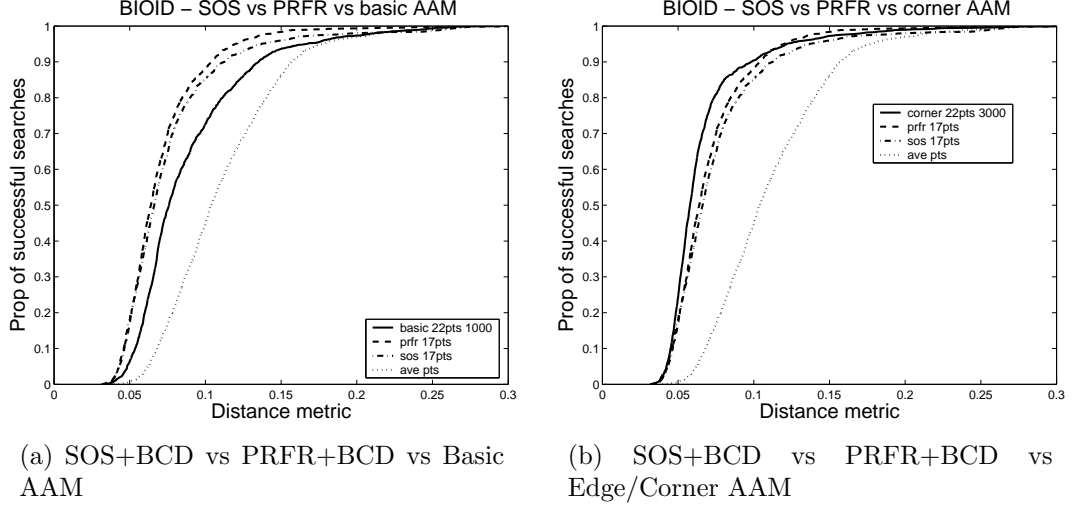


Figure 7.11: Comparing PRFR, SOS and AAM point to point error m_{e17} when searching the BIOID test set

AAM search at all proximity thresholds (m_{e17}). For example with $m_{e17} = 0.15$, the SOS achieves a success rate of 96% and the PRFR 98%, compared to the basic AAM only managing 92%. All three local search methods perform much better than the average point prediction with only find 85% of faces.

When switching to the edge/corner sampling the AAM search compares much more favourably with the SOS and PRFR methods. Figure 7.11(b) shows that with a proximity threshold of $m_{e17} = 0.15$ the edge/corner AAM is successful for 97% of faces, compared with SOS 96% and PRFR 98%. Therefore the PRFR is the most successful method at a proximity threshold of $m_{e17} = 0.15$.

However, the shape of the curves in Figure 7.11(b) show that the edge/corner AAM is the most successful method if the proximity threshold is set tighter, i.e. any value of $m_{e17} < 0.1$. The PRFR is only more successful if $m_{e17} > 0.1$. The SOS method is worse than the PRFR and edge/corner AAM at all values of m_{e17} . This indicates that the PRFR method is the most robust method in the sense that the predicted points are approximately correct most of the time. The edge/corner AAM is slightly less robust, but in many cases able to predict feature points more accurately than the PRFR.

7.6.2 Combined PRFR and AAM Search

Section 7.6.1 suggests that the PRFR search method is the most robust local search method, but the edge/corner AAM is the most accurate. This suggests that the two methods could be combined. For example the PRFR method can be applied to predict the approximate location of the feature points and the edge/corner AAM search applied to further improve the accuracy. Figure 7.12 shows the search accuracy of average point prediction, PRFR, edge/corner AAM search initialised using average points and edge/corner AAM search initialised using PRFR points. The best performing method is the PRFR+AAM search.

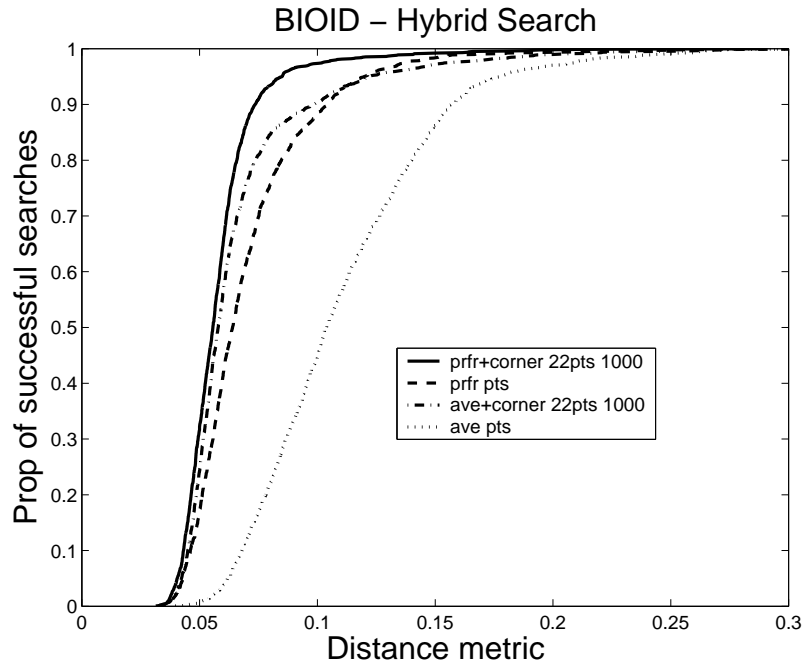


Figure 7.12: PRFR+AAM Search Error m_{e17} when searching the BIOID test set

In Figure 7.12 using a proximity threshold of $m_{e17} = 0.1$, the average points are accurate enough for only 48% of faces. When using the edge/corner AAM (22pts +1000 pixels) starting from the average points the success rate increases to 90%. When applying the PRFR algorithm the success rate is 87% (with $m_{e17} = 0.1$). However

when applying the PRFR algorithm then initialising the AAM with these points, the success rate jumps to 97%. At all values of m_{e17} the PRFR+AAM refinement search is more successful than the initial PRFR point predictions.

Therefore AAM edge/corner search works very well, but only given a good starting position provided by the PRFR algorithm. It appears that the average points are not accurate enough, which means that the AAM finds false minima instead of the true solution in many cases. The PRFR+edge/corner AAM hybrid method gives the best automatic feature point predictions that can be achieved using the methods in this thesis.

7.6.3 Individual Feature Accuracy

Figure 7.12 shows that the PRFR+edge/corner AAM search improves on the PRFR method alone, using the m_{e17} proximity measure. However, it is also possible to analyse the point to point errors for individual features points. Figure 7.13 plots the average point to point errors (m_{e1}) over all images in the BIOD test set for each individual feature, using PRFR and PRFR+AAM.

Figure 7.13 shows that the mouth and nose features are the points that are most improved by the application of the AAM search to the initial PRFR points. For example the lower lip error m_{e1} is improved $10.14\% \rightarrow 7.46\%$ and the nose tip is improved $9.35\% \rightarrow 7.04\%$. However the improvement in m_{e1} is much less for other features such as the right eye $4.37\% \rightarrow 4.12\%$

The average point to point error m_{e17} over all feature points plotted in Figure 7.13 is 7.07% using PRFR, but reduces to 5.85% when refining feature locations with the edge/corner AAM search. The improvement in m_{e17} using PRFR+AAM relative to PRFR, can be shown to be mainly due to the mouth and nose regions. For example the improvement in point to point error of the nose+mouth region (i.e. nostrils, nose tip + upper/lower lip + mouth corners) is $7.97\% \rightarrow 5.86\%$, whilst the improvement

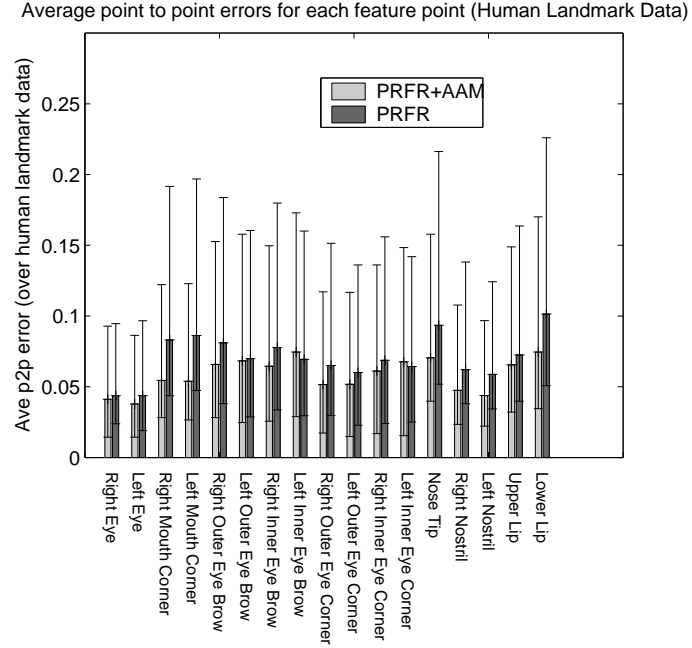


Figure 7.13: Point to point error m_{e1} for each feature when performing PRFR and PRFR+AAM searches on the BIOD test set

in the eye region (i.e. eye pupils, eye corners + eye brows) is only $6.44\% \rightarrow 5.84\%$.

7.6.4 PRFR+AAM Timings

The search times using the PRFR + edge/corner AAM are shown in Table 7.2.

Event	Time
Global search	~400ms
Local BCD search	~150ms
PRFR	~700ms
Local Edge/Corner 22pt 1000 pixel search	~400-450ms
Total	~1650-1700ms

Table 7.2: Time to perform PRFR+AAM search on a single BIOD image, using a 500Mhz PII processor

The total search time per image is quite large ~ 1.5 secs. This could probably be improved by making some efficiency improvements to the PRFR search (see Section 6.4.5). However the PRFR+AAM method in its current form could still search

several frames per second, using more modern hardware.

7.6.5 PRFR+AAM Search Conclusions

Figure 7.12 shows that the PRFR + edge/corner AAM search is by far the best search method presented in this thesis. The PRFR+AAM method outperforms both the PRFR algorithm described in Section 6.4 and the edge/corner AAM search (described in Section 7.3.3) initialised using the average points.

This suggests that the AAM search requires a very good initialisation to obtain accurate results. The average points predicted by the global face region are not accurate enough to initialise the AAM in many cases. However, when using the PRFR method to predict feature locations and then seeding the edge/corner AAM with these points, very accurate results can be obtained.

However, this hybrid method uses several algorithms sequentially, so is computationally expensive. Using a 500MhZ PII processor and applying the hybrid algorithm to the BIODID images, the current implementation takes ~ 1.5 secs to locate facial features.

7.7 Comparison with Other Published Results

Jersorsky *et al.* [43] first introduced the BIODID data set and published results on the eye pupil finding accuracy of their algorithm, which uses a face matching method based on the Hausdorff distance followed by a Multi-Layer Perceptron (MLP) eye finder (see Section 2.5). Jesorsky *et al.* also present eye location accuracy results on the XM2VTS data set. Recently Hamouz *et al.* [39] also presented eye finding results on the BIODID and XM2VTS test sets using a feature based face detection method (see Section 2.4 for more details of this approach). These two methods can be compared with the PRFR+AAM algorithm for the task of eye pupil detection.

Jersorsky *et al.* [43] use a distance measure for each search, which records the maximum point to point error over both eye point predictions, normalised by the known inter-ocular separation. We refer to this distance measure as \hat{m}_{e2} . It is similar to the d_{eyes} distance measure used in Chapter 3 of this thesis to compare face detection algorithms (see Section 3.7.1). Figure 7.14(a) plots \hat{m}_{e2} for the first two sessions of the XM2VTS data set [59], which consists of 1180 images. Similarly Figure 7.14(b) plots \hat{m}_{e2} for both methods on the BIOID data set.

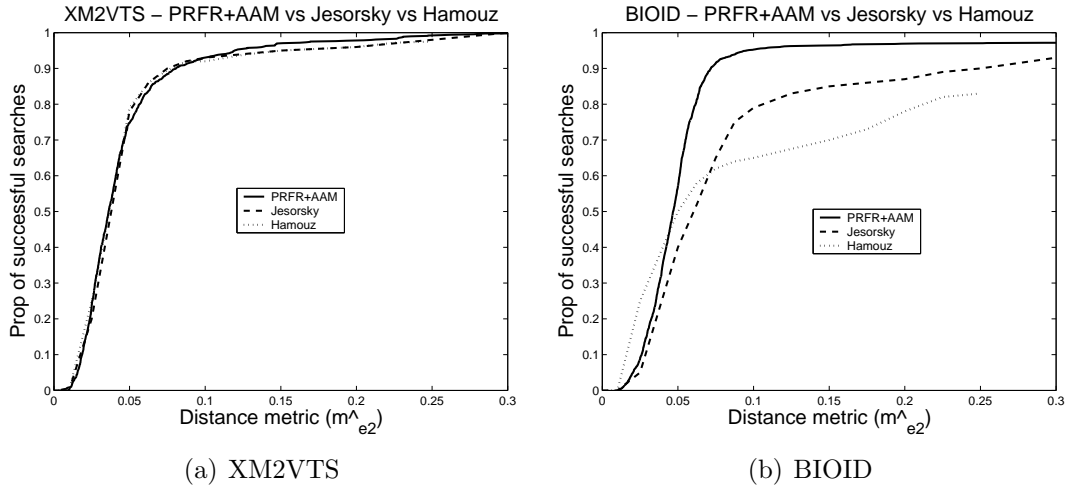


Figure 7.14: Eye pupil finding comparison on the XM2VTS [59] and BIOID test sets [43]

The graphs in Figure 7.14 show slightly different results from other graphs in this chapter. For example:-

- The distance measure \hat{m}_{e2} is plotted for all images, even if the global face detector fails.
- An artificial border is created around each BIOID image by replicating the edge pixels. This enables the entire BIOID image set to be searched, because the Boosted Cascade Face Detector is able to find faces lying next to the edge of the image.

These changes allow a direct comparison between the multi-stage approach described

in this thesis and the published results of Jersorsky *et al.* [43] and Hamouz *et al.* [39].

Figure 7.14(b) shows that when applied to the BIODID images, the PRFR+AAM is more successful than the Jesorsky method for all values of \hat{m}_{e2} . For example with $\hat{m}_{e2} = 0.1$, the PRFR+AAM search finds 96% of faces successfully compared to 79% using the Jesorsky approach. The Hamouz* method is more likely to find eye pupils very accurately (e.g. $\hat{m}_{e2} < 0.05$), but is not very robust, sometimes failing to find the face completely and is the worst performing method on the BIODID data set for $\hat{m}_{e2} > 0.1$.

However the results are very different for the XM2VTS data set. Figure 7.14(a) shows that the accuracy of all three eye finding methods are very similar on the cleaner XM2VTS images. With $\hat{m}_{e2} = 0.1$, the Hausdorff+MLP search, Hamouz approach and PRFR+AAM achieve a success rate of around 93% and give very similar performance for all values of \hat{m}_{e2} . This indicates that the Hausdorff+MLP and Hamouz methods work well on relatively clean images under controlled conditions (e.g. the XM2VTS data set), but are less successful on the more complicated BIODID data set. The multi-stage Boosted Cascade Face Detector + PRFR+AAM search can find eye pupils reliably on both data sets.

7.8 Comparison with Human Landmarking

In order to estimate the theoretical limit on the accuracy with which facial features can be located, ten volunteers were asked to hand label ten images from the BIODID data set, shown in Appendix I. For each image the ground truth for each of the 17 feature points was approximated by taking the average of the human landmark locations. Figure 7.15 shows the mean positional errors (m_{e17}) of human landmarking and PRFR point prediction for each of the ten images. For each image the errors are

*Here we take the best face match only, *not* the top 30 hypotheses, which are used by Hamouz *et al.* [38] for face verification.

constructed as follows.

1. Human Error - Mean positional error (m_{e17}) over all 17 features and all 10 volunteers for each image relative to the average human landmark locations. In addition, error bars are plotted to show the range of human landmark errors (m_{e17}) for each image.
2. PRFR+AAM Error - Mean positional error (m_{e17}) over all features for each image, relative to the average human landmark locations.

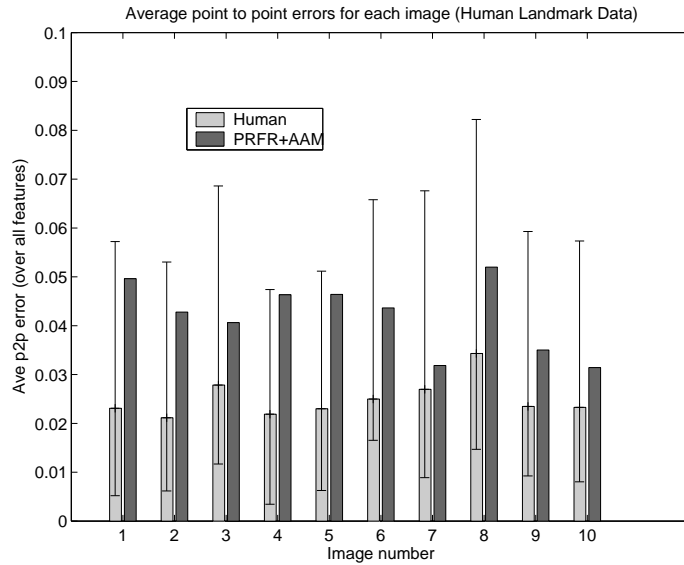


Figure 7.15: Bar chart to show average point to point error (m_{e17}) for each of the 10 test images, using human landmarks and points predicted using PRFR+AAM search.

Figure 7.15 shows that the PRFR+AAM error is worse than the average human error for all 10 images. However, in some cases a human landmarker is less accurate than PRFR+AAM using the m_{e17} distance measure, relative to the average human landmarks. Therefore the PRFR+AAM search behaves like a badly performing human landmarker.

Generally the average human error is evenly distributed over all 10 images in Figure 7.15. The human landmark reproducibility is at its worst for images 3, 7 and 8

(see images in Appendix I). However, it is not clear why human landmarking is less consistent for these images.

The PRFR+AAM error in Figure 7.15 is also variable between images. The case where the PRFR+AAM search deviates most from the average human landmark ground truth is image 8 (see Appendix J). This is mainly due to difficulties in locating the upper/lower lip points and also the outer end of the right eye brow. The PRFR+AAM error is least for images 7,9 and 10 (see Figure 7.15), however is not clear why this is the case.

The differences in search error between different facial features are shown in Figure 7.16 for both PRFR+AAM point prediction and human landmarking. More explicitly the average feature errors in Figure 7.16 are computed as follows:-

1. Human Error - Mean positional error (m_{e1}) over all 10 images and all 10 volunteers for each feature relative to the average human landmark locations.
2. PRFR+AAM Error - Mean positional error (m_{e1}) over all 10 images relative to the average human landmark locations.

Figure 7.16 shows that for all features, the average human positional error is less than the average PRFR+AAM point prediction error, over the 10 test images. The PRFR+AAM method is particularly error prone for the lower lip feature over the 10 images. For the eye and eye brow regions the average mean errors for the PRFR+AAM and human landmarking are generally closer than for other features. The range of point to point errors overlaps for many eye and eye brow features in Figure 7.16.

However, the average positional error (m_{e17}) of human labelling over the 10 images is 2.50% error [†] versus 4.20% using PRFR+AAM. The average mean positional error

[†]This agrees with the average error of human landmarking quoted by Chen *et al.*[5], who label 30 FERET images using 25 human landmarkers

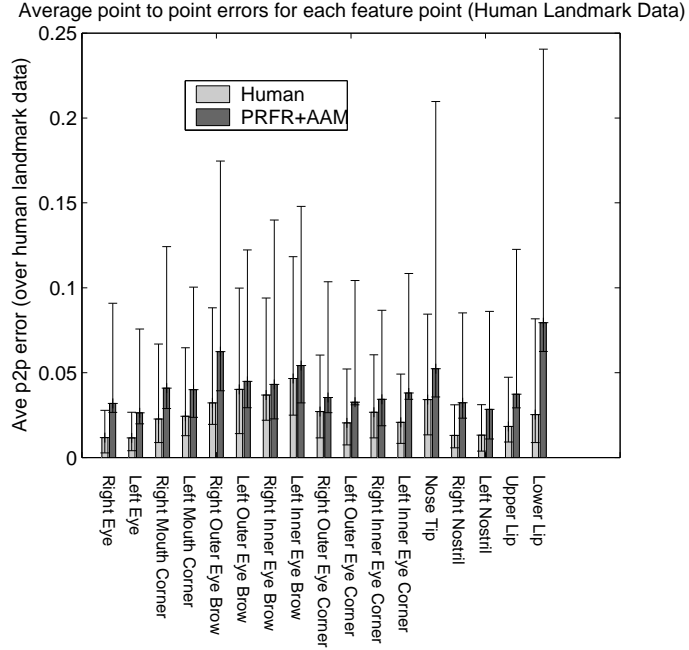


Figure 7.16: Bar chart to show average point to point error for various features, using human landmarks and points predicted using PRFR+AAM. The error bars show the range of positional error m_{e1} for each feature.

(m_{e17}) over the whole BIODID test set is 5.85% using PRFR+AAM. This indicates that PRFR+AAM performs better on the ten image subset relative to the whole BIODID data set, but still worse than human landmarking. Therefore automatic landmark placement using PRFR+AAM cannot replace a human placing landmarks manually, but can give a sensible markup in many cases.

7.9 Conclusions

This chapter gives an overview of the Active Appearance Model algorithm due to Cootes *et al.* [6] and compares different formulations of the AAM when searching the BIODID test set. The most critical variation is the edge/corner texture sampling method recently introduced by Scott *et al.* [82]. This texture sampling method is found to increase the accuracy of search compared to the original normalised texture processing described by Cootes *et al.* [6].

The edge/corner AAM search is then compared with the shape constrained feature detection algorithms described in Chapter 6. It is found that the edge/corner AAM is more accurate than the PRFR algorithm for many faces, however the edge/corner AAM is sometimes less accurate due to poor initialisation. The basic AAM algorithm is found to perform much worse than both the PRFR and SOS algorithms described in Chapter 6.

The main conclusion of this chapter is that very accurate feature localisation can be achieved if a hybrid approach of methods described in this thesis is adopted. The PRFR algorithm can be used to predict approximate feature locations. These points can then be refined using the edge/corner AAM search. This PRFR+AAM method is found to easily out perform both the initial PRFR search and the AAM search initialised using average points. Also the PRFR+AAM search is shown to give superior results to previous published methods applied to the BIODID test set (see Section 7.7).

Finally the PRFR+AAM method is compared with human landmarking. Humans are able to mark faces with an average error of 2.50%. The average error for automatic landmarking using PRFR+AAM is found to be 4.20% over the same small subset of the BIODID data set. Therefore automatic feature location is worse than human landmarking, but is comparable.

Chapter 8

Discussion and Conclusions

This thesis has described methods for the automatic detection of facial features. This chapter gives an overview of the main conclusions and suggests future avenues of research.

8.1 Summary of Thesis

Chapter 3 : Face Detection Methods

Four face detection methods are implemented and described, namely the Boosted Cascade Detector (BCD), Orientation Map Detector (OMD), Normalised Correlation Detector (NCD) and Linear Profile Detector (LPD). Of these four methods, the Boosted Cascade Detector due to Viola and Jones [92] was found to be the most successful face detector. Generally the BCD was more reliable than the OMD, which was more reliable than the NCD and LPD methods. The difference in performance between the BCD and OMD was particularly acute when tested on the difficult CMU test set.

Chapter 4 : Boosted Cascade Detector Experiments

The build parameters for the Boosted Cascade Detector were investigated in more

detail. The most critical parameters were found to be the size of the training set and the number of features retained in each level of the cascade. The template resolution was shown to be non-critical, but the template region was important. Modelling a large face region, which includes the outline of the face, gave the best performance. The whole face detector was much more reliable than modelling an individual feature, such as the right eye, which has much less local structure.

Chapter 5 : Shape Modelling

Statistical shape models were described and an example model presented which uses 14 parameters to represent the shape variation of 20 facial features, spanning 128 identities. Shape modelling techniques were discussed, because shape is a reliable constraint which can be used to distinguish between plausible and implausible combinations of candidate features.

Chapter 6 : Shape Constrained Feature Detection

Three feature detection algorithms were described (CSS, SOS, PRFR), which combine local feature detection with shape constraints. All three shape constrained algorithms performed much better than unconstrained feature detection. The best performing method was found to be the PRFR algorithm with boosted cascade detectors. However the SOS algorithm was found to perform better than the PRFR method if weaker detectors such as orientation maps and normalised correlation are used. Both the PRFR and SOS methods found it relatively easy to locate eye features (e.g. eye pupils, eye corners and eye brows), but were less accurate at locating eye and nose features (e.g. nose tip, nostrils, mouth corners and lower/upper lip).

Chapter 7 : Active Appearance Models

Different variations of the Active Appearance Model (AAM) were evaluated on the BIOID data set. The most critical variation was found to be the texture sampling method used to model the face region. Edge/corner sampling recently developed by Scott *et al.* [82] was found to give improved results relative to normalised texture sampling, used in the original AAM approach [6].

The edge/corner AAM local search was compared with the PRFR algorithm described in Chapter 6. It was found that when using the average points predicted by the face detector to initialise the search, the AAM sometimes found points more accurately than the PRFR, but was less reliable, sometimes finding false minima far away from the correct feature locations. Therefore the AAM search was later initialised using the PRFR point predictions as starting points. This formulation was found to give significant improvements over both AAM search using average points and PRFR point prediction.

The PRFR+AAM search was compared with human landmarking and found to give a search error* of 4.20% on a small subset of the BIOID data set, for which the standard deviation of human error was 2.50%. Therefore PRFR+AAM search was not as good as human landmarking, but was comparable.

8.2 Discussion

One conclusion of this thesis is that when automatically labelling facial features the search method should be coarse-to-fine. The global face template detection methods described in Section 2.2 are generally more robust than the direct feature detection methods discussed in Section 2.4. Therefore it is sensible to use the classification power of whole face detection methods (the Viola-Jones “Boosted Cascade Detector ” is used in this thesis) and leave the accurate localisation of individual features to a later stage.

However the task of locating facial features, given a box around the face indicating the correct location, scale and orientation is still a non-trivial task. In this thesis, feature detectors based on the Viola-Jones Boosted Cascade Detector were used to locate facial points within the candidate face region. Unfortunately these detectors

*Here search error is the average point to point error over all 17 feature points, normalised by the known inter-ocular separation, see Section 6.1.4

were found to be unreliable, due to the lack of consistent local structure around (relatively) salient points on the face, such as the nostrils and mouth corners. The localisation accuracy of feature detectors were only found to improve when shape constraints were applied. The shape constraints couple the small templates used to model each feature and therefore make the search more robust. Three methods of combining shape and feature detection were presented and shown to be reasonably successful.

The AAM approach to local face search was discussed in Chapter 7. The AAM algorithm is fundamentally different from the local feature detector methods. The AAM is based on a statistical model of face variation, which already combines shape and texture in an appearance model of the whole face. The iterative update rule of the AAM is designed to converge to the correct solution for an unseen face and thus enable accurate feature localisation. However, it is unclear whether the AAM formulation is theoretically better than the shape constrained detection algorithms presented in Chapter 6.

The empirical experiments on the AAM indicated that the best formulation is the edge-based AAM (due to Scott *et al.* [82]). Edge-based texture sampling was shown to give superior results to the original basic AAM. The edge-based AAM gave similar performance to the PRFR feature based method when initialised with the average face points in the candidate face region. The edge-based AAM is formulated to be more sensitive to corner and edge features, so perhaps it is not surprising that this formulation and the more explicit feature detection of the PRFR method gave similar results.

However the best results were achieved using a hybrid approach, which used PRFR to seed the edge-based AAM. This indicates that the local search of the AAM is very robust, but when initialised too far away from the correct solution the AAM is quite likely to fall into a false minima. The robustness of the PRFR method can be used to nudge the AAM closer to the correct solution. This hybrid local search provided

the best feature localisation results in this thesis.

The final formulation was found to be comparable with human landmarking (over a small test set), which is surprising as such comparisons are usually highly unflattering to the automated method. However, there is room for improvement, as the average human variation is still less than the error of the computer landmarking relative to the average human annotated points.

8.3 Future work

Improvements to AAM and PRFR

The AAM models used in this thesis could potentially be improved by adding more hand labelled points to the training set. The models built in this thesis use 22 point models, however the original AAM approach [6] used a 68 point scheme to markup the face. An AAM built with 68 points could model the outline of the jaw more accurately and potentially give more accurate results. The PRFR could also be made more computationally efficient by trimming the size of the histogram images. More powerful feature detectors may be developed for each feature, which could also increase overall search accuracy.

Incorporating a multi-view face detector

A multi-view face detector, which was able to reliably find faces rotated in and out of plane, would improve the performance of the system. This is not only because non-upright faces could be detected, but also because the pose information from such a detector could be used to further constrain the search for individual features. In cases of extreme face rotation separate local feature and shape models could be used to find features, even without non-frontal views. However, multi-view face detection is generally less reliable than frontal face detection (see literature review Section 2.3).

Use of PRFR+AAM to improve Face Recognition

The success rate of face recognition algorithms are highly affected by the accuracy of face detection [55]. Therefore it would be interesting to assess the improvement in face recognition that may be possible using PRFR+AAM to locate internal face features to register face images, compared to just using a global face detector with no feature detection.

Face Candidate Verification

The methods described in this thesis locate facial feature points assuming the face detector has obtained a correct match. However this thesis did not consider if the PRFR+AAM search could also be used to verify the presence or absence of a human face, within the candidate region returned by the face detector. If a reliable quality of fit measure could be calculated this could be used to check that a face has indeed been found by the face detector.

Use of PRFR+AAM for Tracking

PRFR+AAM could be extended for use in tracking. Here the starting point from the previous frame in a sequence would be used to initialise the search. A quality of fit threshold would have to be defined to enable the global search to restart if local search failed. This would greatly improve the efficiency of PRFR+AAM when used to search video data.

Semi-Automatic Building Schemes

One of the problems with the algorithms described in this thesis is the large amount of training data required to build the face models and time taken to manually landmark each training image. If automatic model building methods could be developed this would make application of these techniques much less labour intensive. Semi-automatic techniques could perhaps be used to select some of the model parameters, for example the regions chosen to build each feature detector. If these regions could be chosen in a more automatic way then perhaps search results could be improved.

Apply techniques to non-face objects

The BCD+PRFR+AAM search could be applied to other objects. For example images of cars, with consistent internal features, e.g. number plates, headlights etc. The techniques could also be applied to other deformable objects, for example medical images of internal organs.

Extension to 3D

The shape constrained algorithms could also be used in three dimensions. For example with MRI medical image data. Feature detectors would have to be three dimensional, but once built could easily be constrained using 3D statistical shape models. AAMs have already been extended to three dimensions [96][60][86]. However, one problem with 3D is likely to be the availability of a large enough marked up training set.

8.4 Final Statement

A coarse-to-fine algorithm has been developed, which first calculates the approximate position and scale of the human face, then applies shape constrained local feature detectors. An improved version of the AAM, which is tuned to edge/corner texture, is then applied to refine the final feature locations. This multi-stage approach will be useful for many computer vision applications, which require the accurate location of facial features, such as face recognition, expression recognition and automatic landmarking.

Appendix A

Aligning Two 2D Shapes

Given two 2D shapes, \mathbf{x} and \mathbf{x}' , we wish to find the similarity transform $T(\mathbf{x})$ which minimises the least squares distance between the two shapes, as follows.

$$E = |T(\mathbf{x}) - \mathbf{x}'|^2 \quad (\text{A.1})$$

The 2-D similarity transform $T(\mathbf{x})$ has general form

$$T \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} a & -b \\ b & a \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} t_x \\ t_y \end{pmatrix} \quad (\text{A.2})$$

Without loss of generality both shapes can first be translated to the origin. This is achieved by computing the centre of gravity (CoG) of each shape.

$$\mathbf{x}_c = \left(\frac{\sum x_i}{n}, \frac{\sum y_i}{n} \right) \mathbf{x}'_c = \left(\frac{\sum x_i}{n}, \frac{\sum y_i}{n} \right) \quad (\text{A.3})$$

With the CoG of both shapes at the origin $t_x = 0$ and $t_y = 0$ in Equation A.2. The problem of then finding the correct scale and rotation reduces to minimising the

following expression

$$\begin{aligned} E(a, b) &= |T(\mathbf{x}) - \mathbf{x}'|^2 \\ &= \sum_{i=1}^n (ax_i - by_i - x'_i)^2 + (bx_i + ay_i - y'_i)^2 \end{aligned} \quad (\text{A.4})$$

Differentiating with respect to both a and b and equating to zero gives

$$\begin{aligned} \sum_{i=1}^n ax_i^2 - by_i^2 - x_i x'_i - y_i y'_i &= 0 \\ \sum_{i=1}^n bx_i^2 - by_i^2 - x_i y'_i + y_i x'_i &= 0 \end{aligned} \quad (\text{A.5})$$

This implies

$$\begin{aligned} a &= (\sum_{i=1}^n x_i x'_i + y_i y'_i) / (\sum_{i=1}^n x_i^2 + y_i^2) = \mathbf{x} \cdot \mathbf{x}' / |\mathbf{x}| \\ b &= (\sum_{i=1}^n x_i y'_i - y_i x'_i) / (\sum_{i=1}^n x_i^2 + y_i^2) = (\sum_{i=1}^n x_i y'_i - y_i x'_i) / |\mathbf{x}| \end{aligned} \quad (\text{A.6})$$

Given a, b, \mathbf{x}_c and \mathbf{x}'_c a shape \mathbf{x} can be approximately mapped to a shape \mathbf{x}' as follows

$$\mathbf{x}' \simeq \mathbf{x}'_c + T(\mathbf{x} - \mathbf{x}_c) \quad (\text{A.7})$$

Appendix B

Weighted fitting of PCA parameters

In some cases it is necessary to compute the PCA parameters \mathbf{b} with respect to a weighted set of points, where weights indicate the relative importance of the model fit to each point. We wish to find \mathbf{b} , in the following equation.

$$\mathbf{y} \approx \bar{\mathbf{x}} + \Phi \mathbf{b} \quad (\text{B.1})$$

Here \mathbf{y} represents a given shape, $\bar{\mathbf{x}}$ the PCA mean and Φ the PCA eigenvector matrix. The weights are introduced using a diagonal weight matrix \mathbf{W} . Then satisfying Equation B.1 can be reformatted as selecting parameter \mathbf{b} to minimise the following expression.

$$|\mathbf{W} (\mathbf{y} - \bar{\mathbf{x}} - \Phi \mathbf{b})|^2 \quad (\text{B.2})$$

Setting $\mathbf{v} = \mathbf{y} - \bar{\mathbf{x}} - \Phi \mathbf{b}$ and differentiating the expression $\mathbf{v}^T \mathbf{W}^2 \mathbf{v}$ with respect to \mathbf{b} gives the following.

$$\begin{aligned}
 \frac{d}{d\mathbf{b}} (\mathbf{v}^T \mathbf{W}^2 \mathbf{v}) &= \frac{d\mathbf{v}}{d\mathbf{b}} \times \frac{d(\mathbf{v}^T \mathbf{W}^2 \mathbf{v})}{d\mathbf{v}} \\
 &= \mathbf{\Phi}^T \times -2\mathbf{W}^2 \mathbf{v} \\
 &= -2\mathbf{\Phi}^T \mathbf{W}^2 (\mathbf{y} - \bar{\mathbf{x}} - \mathbf{\Phi} \mathbf{b})
 \end{aligned} \tag{B.3}$$

Setting the differential to zero leads to the following matrix equation.

$$\mathbf{\Phi}^T \mathbf{W}^2 \mathbf{\Phi} \mathbf{b} = \mathbf{\Phi}^T \mathbf{W}^2 (\mathbf{y} - \bar{\mathbf{x}}) \tag{B.4}$$

$\mathbf{\Phi}^T \mathbf{W}^2 \mathbf{\Phi}$ is known, as is $\mathbf{\Phi}^T \mathbf{W}^2 (\mathbf{y} - \bar{\mathbf{x}})$, so Equation B.4, can be solved for \mathbf{b} , for example by singular value decomposition (SVD) [68]. Given \mathbf{b} the original points \mathbf{y} can be approximated using Equation B.1.

Note given $\mathbf{W} = \mathbf{I}$ Equation B.4 reduces to the normal unweighted PCA projection $\mathbf{b} = \mathbf{\Phi}^T (\mathbf{y} - \bar{\mathbf{x}})$.

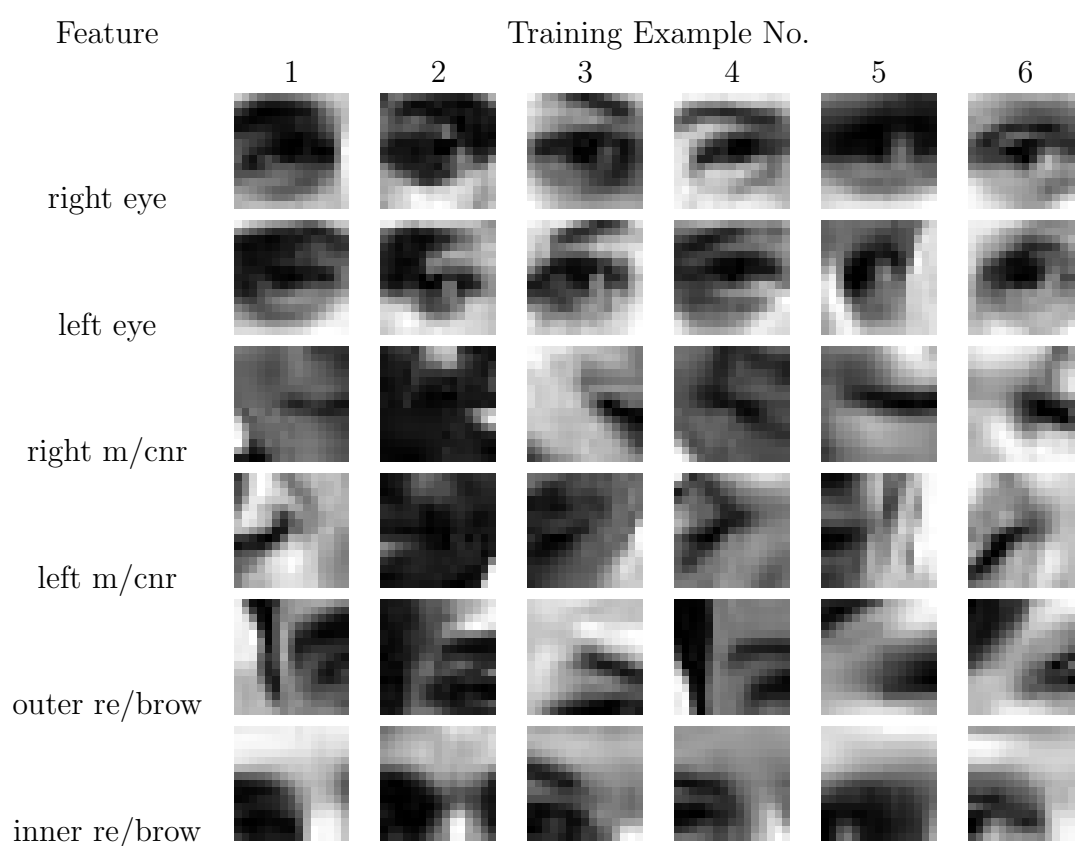
Appendix C

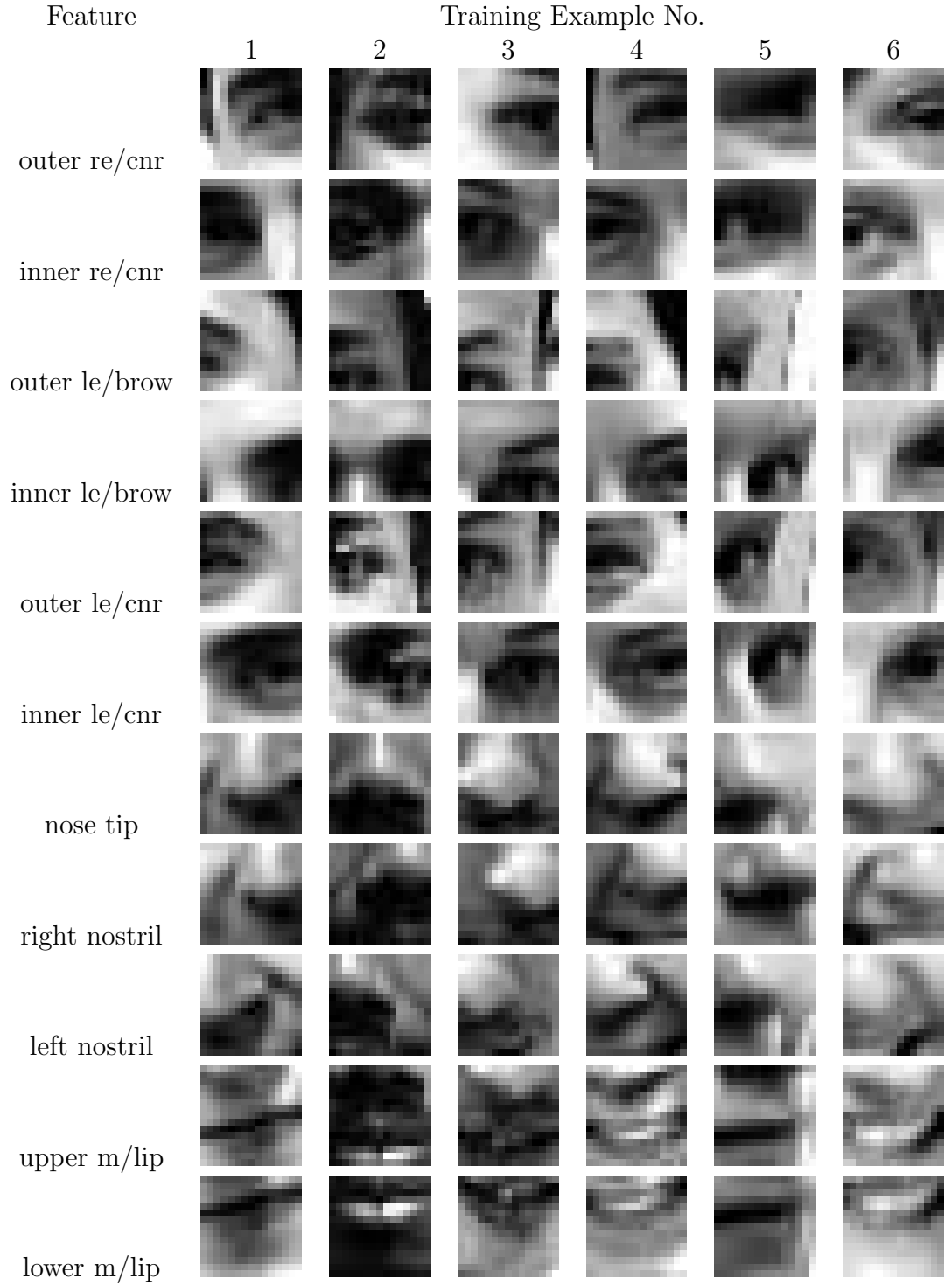
BCD Training Set



Appendix D

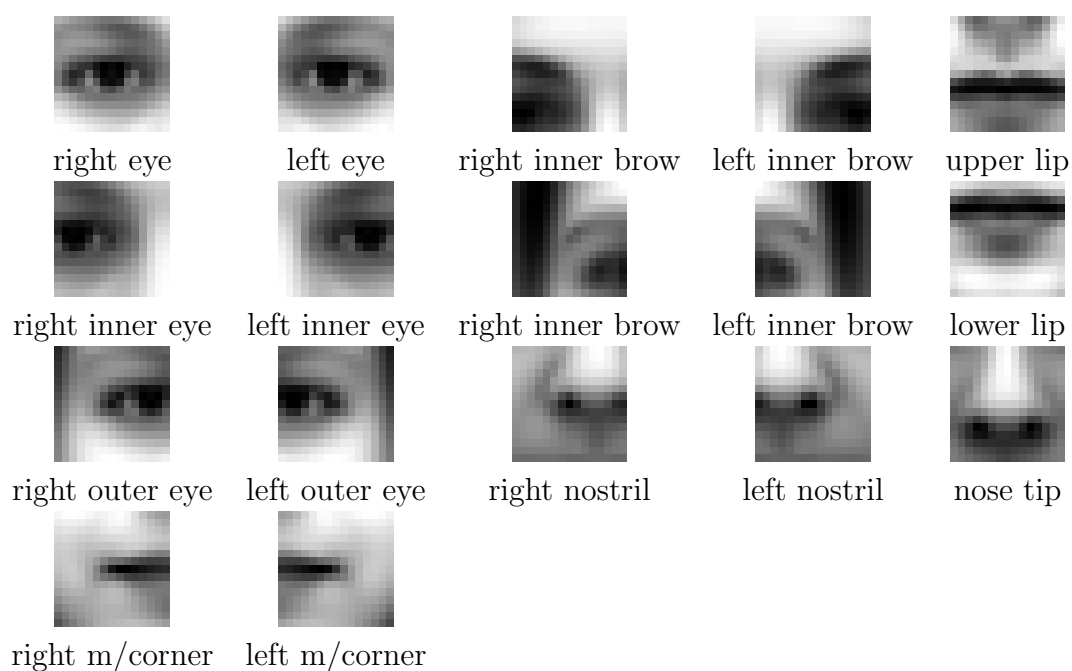
Facial Feature Detector Training Examples





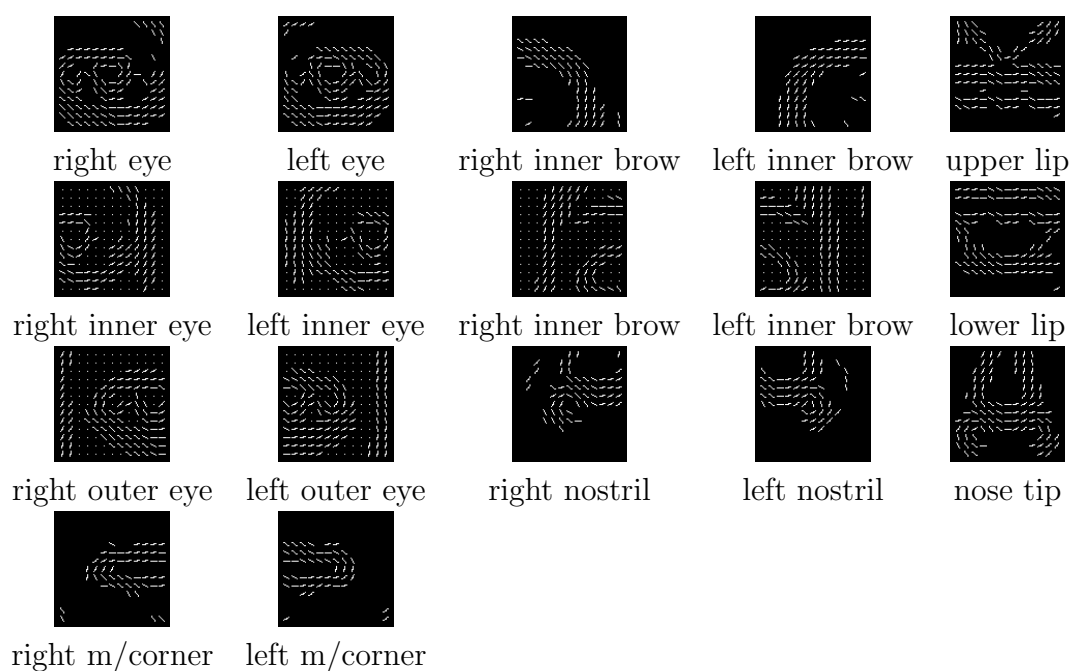
Appendix E

Normalised Correlation Feature Templates



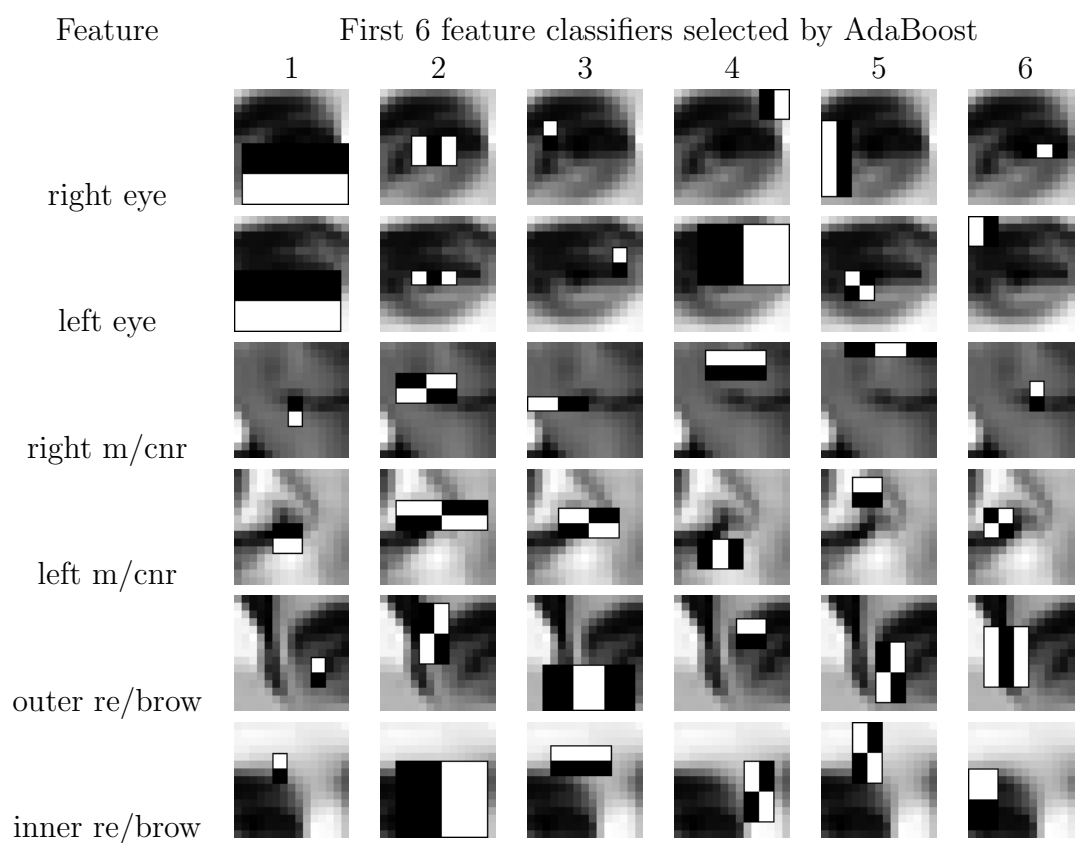
Appendix F

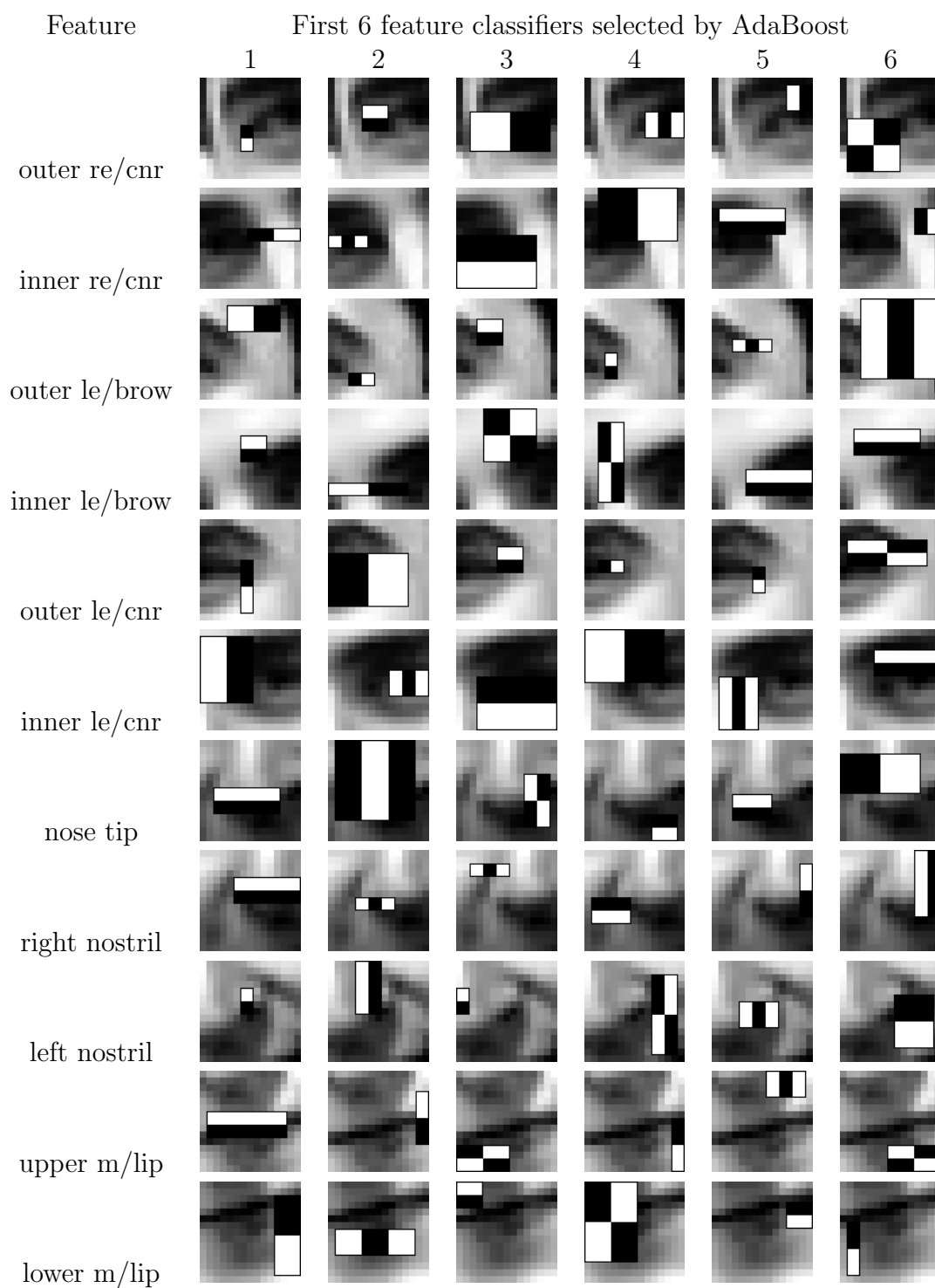
Orientation Map Feature Templates



Appendix G

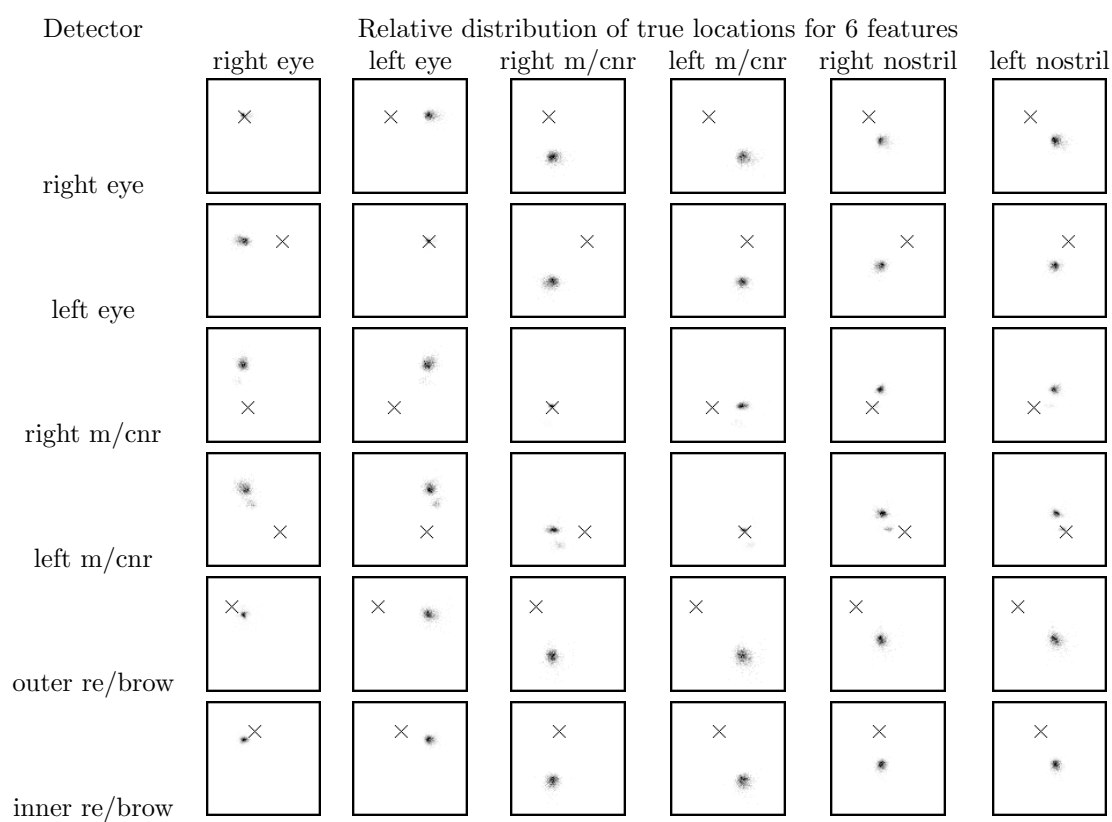
Features selected by AdaBoost

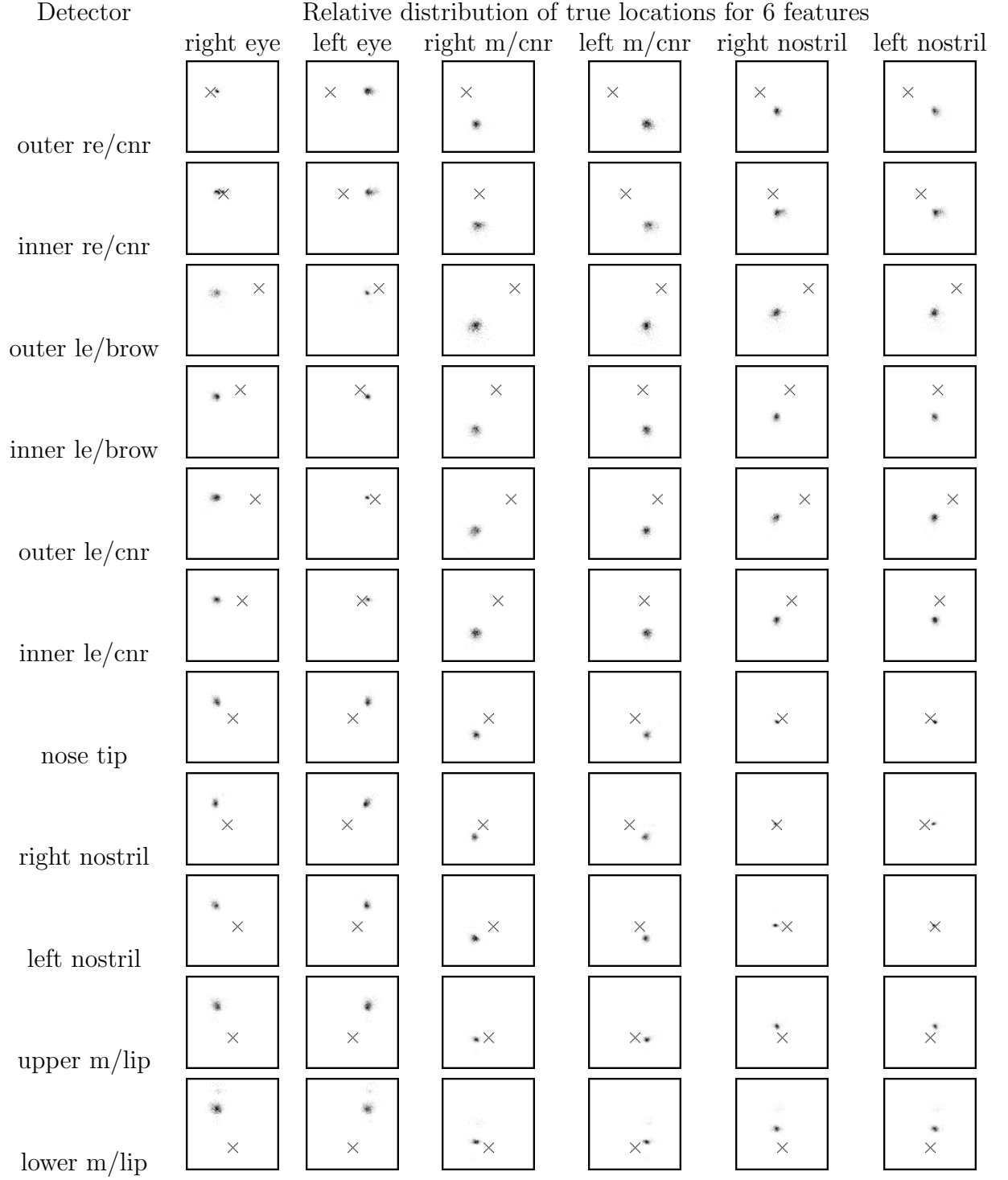




Appendix H

PRFR Histograms





Appendix I

Human Landmark Test Images

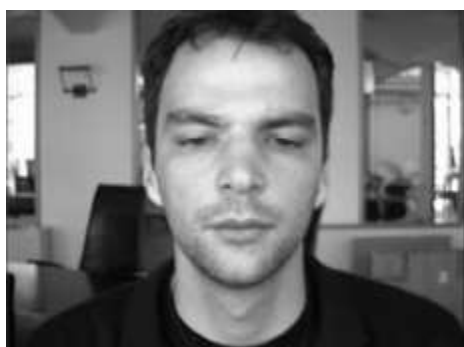


Image 1 (bioid_0000.pgm)



Image 2 (bioid_0010.pgm)



Image 3 (bioid_0050.pgm)



Image 4 (bioid_0090.pgm)



Image 5 (bioid_0250.pgm)



Image 6 (bioid_0329.pgm)



Image 7 (bioid_0499.pgm)



Image 8 (bioid_0579.pgm)



Image 9 (bioid_0609.pgm)



Image 10 (bioid_0669.pgm)

Appendix J

Average Human Landmarks vs PRFR+AAM

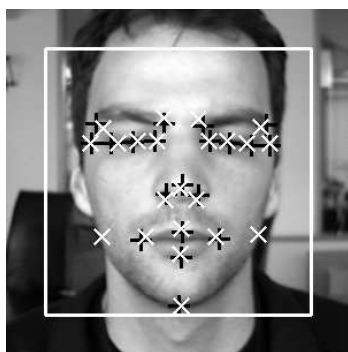


Image 1 ($m_{e17} = 4.96\%$)

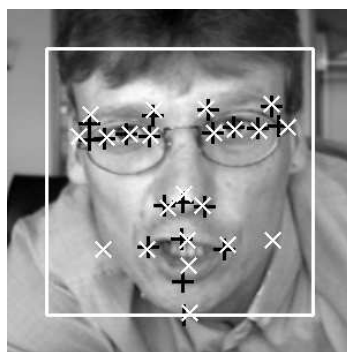


Image 2 ($m_{e17} = 4.28\%$)

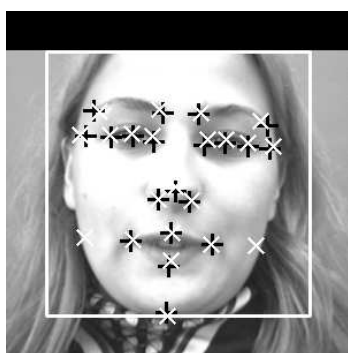


Image 3 ($m_{e17} = 4.06\%$)

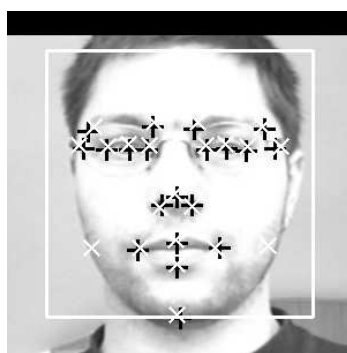


Image 4 ($m_{e17} = 4.63\%$)



Image 5 ($m_{e17} = 4.64\%$)

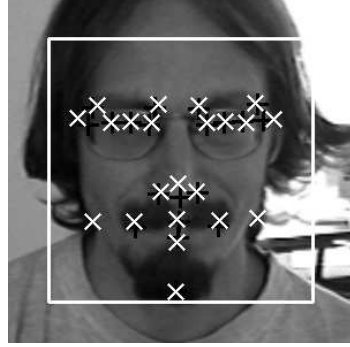


Image 6 ($m_{e17} = 4.36\%$)



Image 7 ($m_{e17} = 3.18\%$)

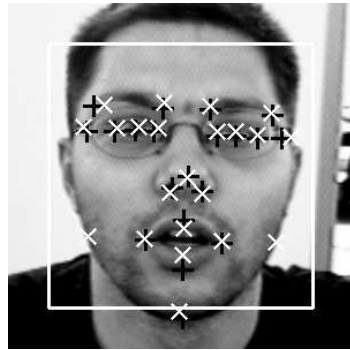


Image 8 ($m_{e17} = 5.20\%$)

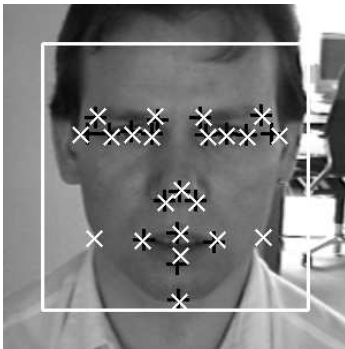


Image 9 ($m_{e17} = 3.50\%$)

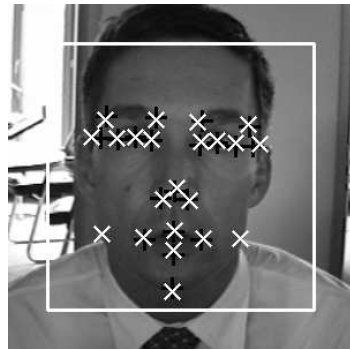


Image 10 ($m_{e17} = 3.14\%$)

Bibliography

- [1] C. Ballard and C. Brown. *Computer Vision*. Prentice-Hall, 1982.
- [2] D. Ballard. Generalising the hough transform to detect arbitrary shapes. *Pattern Recognition*, 3:110–122, 1981.
- [3] L. Breimann, J. Friedman, R. Olshen, and C. Stone. *Classification and Regression Trees*. Wadsworth, 1984.
- [4] M. Burl, T. Leung, and P. Perona. Face localization via shape statistics. In *1st International Workshop on Automatic Face and Gesture Recognition 1995*, Zurich, Switzerland, 1995.
- [5] L. Chen, L. Zhang, L. Zhu, M. Li, and H. Zhang. A novel facial feature point localization algorithm using probabilistic-like output. In *6th Asian Conference on Computer Vision*, Jeju Island, Korea, Jan. 2004.
- [6] T. F. Cootes, G. J. Edwards, and C. J. Taylor. Active appearance models. In H. Burkhardt and B. Neumann, editors, *5th European Conference on Computer Vision*, volume 2, pages 484–498. Springer, Berlin, 1998.
- [7] T. F. Cootes, G. J. Edwards, and C. J. Taylor. Comparing active shape models with active appearance models. In T. Pridmore and D. Elliman, editors, *10th British Machine Vision Conference*, volume 1, pages 173–182, Nottingham, UK, Sept. 1999. BMVA Press.
- [8] T. F. Cootes, C. J. Taylor, D. Cooper, and J. Graham. Training models of shape from sets of examples. In D. Hogg and R. Boyle, editors, *3rd British Machine Vision Conference*, pages 9–18. Springer-Verlag, Sept. 1992.
- [9] T. F. Cootes, G. Page, C. Jackson, and C. J. Taylor. Statistical grey-level models for object location and identification. In D. Pycock, editor, *6th British Machine Vision Conference*, pages 533–542, Birmingham, England, Sept. 1995. BMVA Press.

- [10] T. F. Cootes and C. Taylor. Statistical models of appearance for computer vision. Technical report, Dept of Imaging Science and Biomedical Engineering, Feb. 2001.
- [11] T. F. Cootes and C. J. Taylor. Locating faces using statistical feature detectors. In *2nd International Conference on Automatic Face and Gesture Recognition 1996*, pages 204–211, Los Alamitos, California, Oct. 1996. IEEE Computer Society Press.
- [12] T. F. Cootes and C. J. Taylor. On representing edge structure for model matching. In *Computer Vision and Pattern Recognition Conference 2001*, volume 1, pages 1114–1119, 2001.
- [13] T. F. Cootes, C. J. Taylor, D. Cooper, and J. Graham. Active shape models - their training and application. *Computer Vision and Image Understanding*, 61(1):38–59, Jan. 1995.
- [14] D. Cristinacce and T. Cootes. A comparison of two real-time face detection methods. In *4th IEEE International Workshop on Performance Evaluation of Tracking and Surveillance, Graz, Austria*, pages 1–8, 2003.
- [15] D. Cristinacce and T. Cootes. Facial feature detection using adaboost with shape constraints. In *14th British Machine Vision Conference, Norwich, England*, pages 231–240, 2003.
- [16] D. Cristinacce and T. Cootes. A comparison of shape constrained facial feature detectors. In *6th International Conference on Automatic Face and Gesture Recognition 2004, Seoul, Korea*, pages 375–380, 2004.
- [17] D. Cristinacce, T. Cootes, and I. Scott. A multi-stage approach to facial feature detection. In *15th British Machine Vision Conference, London, England*, pages 277–286, 2004.
- [18] F. C. Crow. Summed-area tables for texture mapping. In *Proceedings of SIG-GRAPH '84*, pages 207–212, 1984.
- [19] F. Dornaika and J. Ahlberg. Face model adaptation for tracking and active appearance model training. pages 559–568, 2003.
- [20] I. Dryden and K. V. Mardia. *Statistical Shape Analysis*. Wiley, London, 1998.
- [21] G. Edwards, A. Lanitis, C. J. Taylor, and T. F. Cootes. Modelling the variability in face images. In *2nd International Conference on Automatic Face and Gesture Recognition 1996*, pages 328–333, Los Alamitos, California, Oct. 1996. IEEE Computer Society Press.

- [22] G. Edwards, C. J. Taylor, and T. F. Cootes. Interpreting face images using active appearance models. In *3rd International Conference on Automatic Face and Gesture Recognition 1998*, pages 300–305, Japan, 1998.
- [23] R. S. Feris, J. Gemmell, K. Toyama, and V. Krüger. Hierarchical wavelet networks for facial feature localization. In *5th International Conference on Automatic Face and Gesture Recognition 2002, Washington, USA*, Washington D.C., USA, May 2002.
- [24] M. Fischler and R. Bolles. Random sample consensus: a paradigm for model fitting with application to image analysis and automated cartography. *Communication Association and Computing Machine*, 24(6):381–395, 1981.
- [25] W. Freeman and E. Adelson. The design and use of steerable filters for image analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13:891–906, 1991.
- [26] Y. Freund and R. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. In *2nd European Conference on Computational Learning Theory*, 1995.
- [27] Y. Freund and R. Schapire. Experiments with a new boosting algorithm. In *Proceedings of the 13th International Conference on Machine Learning*, pages 148–156, 1996.
- [28] B. Fröba and A. Ernst. Fast frontal-view face detection using a multi-path decision tree. In *4th International Conference on Audio- and Video-Based Biometric Person Authentication 2003*, pages 921–928, 2003.
- [29] B. Fröba and A. Ernst. Face detection with modified census transform. In *6th International Conference on Automatic Face and Gesture Recognition 2004, Seoul, Korea*, pages 91–96, 2004.
- [30] B. Fröba and C. Küllbeck. Orientation template matching for face localization in complex visual scenes. In *International Conference on Image Processing ICIP2000*, pages 251–254, 2000.
- [31] B. Fröba and C. Küllbeck. Face detection and tracking using edge orientation information. In *SPIE Visual Communications and Image Processing*, pages 583–594, 2001.
- [32] B. Fröba and C. Küllbeck. Real-time face detection. In *Proceedings of 4th IASTED International Conference on Signal and Image Processing, Kauai, Hawaii*, pages 479–502, 2002.

- [33] B. Fröba and C. Küllbeck. Robust face detection at video frame rate based on edge orientation features. In *5th International Conference on Automatic Face and Gesture Recognition 2002, Washington, USA*, pages 342–347, 2002.
- [34] C. Goodall. Procrustes methods in the statistical analysis of shape. *Journal of the Royal Statistical Society B*, 53(2):285–339, 1991.
- [35] H. Graf, T. Chen, E. Petajan, and E. Cosatto. Locating faces and facial parts. In *1st International Workshop on Automatic Face and Gesture Recognition 1995, Zurich, Switzerland*, 1995.
- [36] G. H. Granlund. In search of a general picture processing operator. *Computer Graphics and Image Processing*, 8:155–173, 1978.
- [37] M. Hamouz. *Feature-based affine-invariant detection and localization of faces*. PhD thesis, Centre for Vision, Speech and Signal Processing, University of Surrey, UK., 2004.
- [38] M. Hamouz, J. Kittler, J. K. Kämäräinen, and H. Kälviöinen. Hypotheses-drive affine invariant localization of faces in verification systems. In *4th International Conference on Audio- and Video-Based Biometric Person Authentication 2003*, pages 276–284, 2003.
- [39] M. Hamouz, J. Kittler, J. K. Kämäräinen, and H. Kälviöinen. Affine-invariant face detection and localization using gmm-based feature detectors and enhanced appearance model. pages 67–72, 2004.
- [40] C. Harris and M. Stephens. A combined corner and edge detector. In *4th Alvey Vision Conference*, pages 147–151, 1988.
- [41] R. Herpers, M. Michaelis, K.-H. Lichtenauer, and G. Sommer. Edge and key-point detection in facial regions. In *2nd International Conference on Automatic Face and Gesture Recognition 1996*, pages 212–217, Killington, Vermont, USA, Oct. 1996.
- [42] E. Hjelm and B. Low. Face detection: A survey. *Computer Vision and Image Understanding*, 83:235–274, 2001.
- [43] O. Jesorsky, K. J. Kirchberg, and R. W. Frischholz. Robust face detection using the hausdorff distance. In *3rd International Conference on Audio- and Video-Based Biometric Person Authentication 2001*, 2001.
- [44] I. Jolliffe. *Principal Component Analysis*. Springer-Verlag, 1986.
- [45] M. J. Jones and P. Viola. Fast multi-view face detection. Technical Report TR2003-96, Mitsubishi Electric Research Labs, 2004.

- [46] V. Krüger. Gabor wavelet networks for object representation. Technical report, University of Maryland, 2001.
- [47] H. Kruppa, M. Casterillon-Santana, and B. Schiele. Fast and robust face finding via local context. In *Fifth IEEE International Workshop on Performance Evaluation of Tracking and Surveillance*, pages –, Nice, France, Oct. 2003.
- [48] T. Leung, M. Burl, and P. Perona. Finding faces in cluttered scenes using random labelled graph matching. In *1st International Workshop on Automatic Face and Gesture Recognition 1995*, Zurich, Switzerland, 1995.
- [49] S. Li, L. Zhu, Z. Zhang, A. Blake, H. Zhang, and H. Shum. Statistical learning of multi-view face detection. In *7th European Conference on Computer Vision*, pages 67–81, 2002.
- [50] Y. Li, S. Gong, and H. Liddell. Support vector regression and classification based multi-view face detection and recognition. In *7th International Conference on Computer Vision*, pages 300–305, 2000.
- [51] Y. Li, S. Gong, J. Sherrah, and H. Liddell. Support vector machine based multi-view face detection and recognition. *Image and Vision Computing*, 22(5):413–427, 2004.
- [52] R. Lienhart, A. Kuranov, and V. Pisarevsky. Empirical analysis of detection cascades of boosted classifiers for rapid object detection. In *25th Pattern Recognition Symposium*, Madgeburg, Germany, 2003.
- [53] R. Lienhart and J. Maydt. An extended set of haar-like features for rapid object detection. In *IEEE International Conference on Image Processing*, pages 900–903, New York, USA, 2002.
- [54] D. Maio and D. Maltoni. Real-time face location on gray-scale static images. *Pattern Recognition*, 33(9):1525–1539, September 2000.
- [55] J. Matas, J. Kittler, J. Luetttin, and G. Maitre. Fast face localisation and verification. In *2nd Conference on Audio and video-base biometric personal verification*, New York, 1999.
- [56] T. Maurer and C. von der Malsburg. Tracking and learning graphs and pose on image sequences of faces. pages 176–181, 1996.
- [57] S. McKenna, S. Gong, and Y. Raja. Modelling facial colour and identity with gaussian mixtures. *Pattern Recognition*, 31(12):1883–1892, 1998.

- [58] S. McKenna, S. Gong, R. Wurtz, J. Tanner, and D. Banin. Tracking facial feature points with gabor wavelets and shape models. In *1st International Conference on Audio- and Video-Based Biometric Person Authentication 1997, Crans-Montan, Switzerland*, 1997.
- [59] K. Messer, J. Matas, J. Kittler, J. Luettin, and G. Maitre. Xm2vtsdb: The extended m2vts database. In *Proc. 2nd Conf. on Audio and Video-based Biometric Personal Verification*. Springer Verlag, 1999.
- [60] S. Mitchell, B. Lelieveldt, R. Geest, J. Schaap, J. Reiber, and M. Sonka. Segmentation of cardiac mr images: An active appearance model approach. *Medical Imaging 2000: Image Processing*, 1, 2000.
- [61] B. Moghaddam and A. Pentland. Face recognition using view-based and modular eigenspaces. In *SPIE*, volume 2277, pages 12–21, 1994.
- [62] B. Moghaddam and A. Pentland. Probabilistic visual learning for object recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(7):696–710, 1997.
- [63] B. Moghaddam, A. Pentland, and T. Starner. View-based and modular eigenspaces for face recognition. In *cvpr94*, volume 2277, pages 84–91, 1994.
- [64] J. A. Nelder and R. Mead. A simplex method for function minimization. *Computer Journal*, 7:308–313, 1965.
- [65] E. Osuna, R. Freund, and F. Girosi. Training support vector machines: An application to face detection. In *Computer Vision and Pattern Recognition Conference 1997*, 1997.
- [66] J. Pearl. *Probabilistic Reasoning in Intelligent Systems*. Morgan Kaufman, San Mateo, California, 1988.
- [67] P. Phillips, H. Wechsler, J. Huang, and P. Rauss. The feret database and evaluation procedure for face recognition algorithms. *Image and Vision Computing*, 16(5):295–306, 1998.
- [68] W. Press, S. Teukolsky, W. Vetterling, and B. Flannery. *Numerical Recipes in C (2nd Edition)*. Cambridge University Press, 1992.
- [69] P. Pudil, J. Novovicova, and J. Kittler. Float search methods in feature selection. *Pattern Recognition Letters*, 15:1119–1125, 1994.
- [70] J. R. Quinlan. *C4.5: programs for machine learning*. Morgan Kaufmann Publishers Inc, 1993.

- [71] M. Reinders, R. Koch, and J. Gerbrands. Locating facial features in image sequences using neural networks. In *2nd International Conference on Automatic Face and Gesture Recognition 1996*, pages 230–236, 1996.
- [72] D. Roth. The snow learning architecture. Technical report no. uiucdcs-r-99-2101, University of Illinois at Urbana-Champaign, Urbana, IL, 1999.
- [73] D. Roth, M. Yang, and N. Ahuja. A snowbased face detector. *Neural Information Processing*, 1(12), 2000.
- [74] H. A. Rowley, S. Baluja, and T. Kanade. Human face detection in visual scenes. In D. S. Touretzky, M. C. Mozer, and M. E. Hasselmo, editors, *Advances in Neural Information Processing Systems*, volume 8, pages 875–881. The MIT Press, 1996.
- [75] H. A. Rowley, S. Baluja, and T. Kanade. Neural network-based face detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(1):23–38, 1998.
- [76] H. A. Rowley, S. Baluja, and T. Kanade. Rotation invariant neural network-based face detection. In *Computer Vision and Pattern Recognition Conference 1998*, pages 38–44, 1998.
- [77] W. Rucklidge. Efficient visual recognition using the hausdorff distance. *Lecture Notes in Computer Science*, 1173, 1996.
- [78] S. Russell and P. Norvig. *Introduction to Artificial Intelligence*. Prentice Hall, 1995.
- [79] R. Schapire. A brief introduction to boosting. In *16th International Joint Conference on Artificial Intelligence*, 1999.
- [80] H. Schneiderman and T. Kanade. Probabilistic modelling of local appearance and spatial relationships for object recognition. In *Computer Vision and Pattern Recognition Conference 1998*, 1998.
- [81] H. Schneiderman and T. Kanade. A statistical approach to 3d object detection applied to faces and cars. In *8th International Conference on Computer Vision*, 2000.
- [82] I. M. Scott, T. F. Cootes, and C. J. Taylor. Improving appearance model matching using local image structure. In *Information Processing in Medical Imaging*, *18th International Conference*, pages 258–269, July 2003.

- [83] T. Shakunaga, K. Ogawa, and S. Oki. Integration of eigentemplate and structured matching for autonomous facial feature detection. In *3rd International Conference on Automatic Face and Gesture Recognition 1998*, 1998.
- [84] R. E. Shapire and Y. Singer. Improved boosting algorithms using confidence-rated predictions. In *11th Annual Conference on Computational Learning Theory*, pages 80–91, 1998.
- [85] M. Sonka, V. Hlavac, and R. Boyle. *Image Processing, Analysis, and Machine Vision*. PWS, London, 1999.
- [86] M. B. Stegmann. *Generative Interpretation of Medical Images*. PhD thesis, Informatics and Mathematical Modelling, Technical University of Denmark, DTU, 2004.
- [87] K. Sung and T. Poggio. Example-based learning for view-based human face detection. A.i. memo 1521, Massachusetts Institute of Technology, 1994.
- [88] K. Sung and T. Poggio. Example-based learning for view-based face detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(1):39–51, 1998.
- [89] M. Turk and A. Pentland. Eigenfaces for recognition. *Journal of Cognitive Neuroscience*, 3(1):71–86, 1991.
- [90] V. Vapnik. *The nature of statistical learning theory*. Springer Verlag, 1995.
- [91] P. Viola and M. Jones. Fast and robust classification using asymmetric adaboost and a detector cascade. In *Neural Information Processing Systems*, volume 14, Vancouver, Canada, 2001.
- [92] P. Viola and M. Jones. Rapid object detection using a boosted cascade of simple features. In *Computer Vision and Pattern Recognition Conference 2001*, volume 1, pages 511–518, Kauai, Hawaii, 2001.
- [93] P. Viola and M. Jones. Robust real-time object detection. In *ICCV Workshop on Statistical and Computation Theories of Vision*, Vancouver, Canada, 2001.
- [94] J. Wiegardt, R. P. Wurtz, and C. von der Malsburg. Gabor-based feature point tracking with automatically learned constraints. *Dynamic Perception*, pages 121–126, 2002.
- [95] L. Wiskott, J. Fellous, N. Kruger, and C. der Malsburg. Face recognition by elastic bunch graph matching. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(7):775–779, 1997.

- [96] C. Wolstenholme and C. Taylor. Wavelet compression of active appearance models. In *2nd International Conference on Medical Image Computing and Computer-Assisted Intervention 1999, Cambridge, UK*, pages 544–554, 1999.
- [97] B. Wu, H. Ai, C. Huang, and S. Lao. Fast rotation invariant multi-view face detection based on real adaboost. *6th International Conference on Automatic Face and Gesture Recognition 2004, Seoul, Korea*, pages 79–84, 2004.
- [98] R. Xiao, L. Zhu, and H. J. Zhang. Boosting chain learning for object detection. In *10th International Conference on Computer Vision*, pages 709–722, Nice, France, 2003.
- [99] K. Yow. *Automatic human face detection and localization*. PhD thesis, University of Cambridge, 1998.
- [100] K. Yow and R. Cipolla. Finding initial estimates of human face location. In *2nd Asian Conference on Computer Vision*, volume 3, pages 514–518, Singapore, 1995.
- [101] K. Yow and R. Cipolla. Towards an automatic human face localization system. In D. Pycock, editor, *6th British Machine Vision Conference*, volume 2, pages 701–710, Birmingham, England, 1995. BMVA Press.
- [102] R. Zabih and J. Woodfill. Non-parametric local transforms for computing visual correspondence. In *3rd European Conference on Computer Vision, Stockholm, Sweden*, volume 2, pages 151–158, 1994.