

Copyright is owned by the Author of the thesis. Permission is given for a copy to be downloaded by an individual for the purpose of research and private study only. The thesis may not be reproduced elsewhere without the permission of the Author.

FEATURE-BASED RAPID OBJECT DETECTION:
FROM FEATURE EXTRACTION TO PARALLELISATION

A thesis presented in partial
fulfilment of the requirements

for the degree of

Doctor of Philosophy in Computer Sciences

at Massey University, Auckland, New Zealand

Andre Luis Chautard Barczak

2007

Contents

1	Introduction	1
1.1	Motivation	1
1.2	Research Scope	2
1.3	Original Contributions	2
1.4	Publications related to this work	3
1.5	Outline of the Thesis	4
2	Literature Review	7
2.1	Computer Vision and Rapid Object Detection	7
2.1.1	Real-time Object Detection, Recognition and Tracking	8
2.1.2	Face Detection	11
2.1.3	Invariant Features and Non-invariant Features	13
2.2	Rapid Object Detection: a review of the Viola and Jones Method	15
2.2.1	Haar-like Features	15
2.2.2	Summed-area Tables	16
2.2.3	Adaboost	18
2.2.4	Cascades of Classifiers	22
2.2.5	Multiresolution Analysis in Feature-based Systems	23
2.2.6	Lighting Conditions	26
2.2.7	Articulated Objects and Motion Features	26
2.3	Parallel Computing	27
2.3.1	Types of parallel machines	28
2.3.2	Beowulf Clusters	29
2.3.3	Cluster Topology	29
2.3.4	Bandwidth and Latency	30
2.3.5	Speedup in parallel computers	31
2.3.6	Message passing paradigm	32
2.3.7	Parallel Image Processing	32
2.4	Conclusions of Literature Review	35

3	Methods and Preliminary Experiments	37
3.1	Implementing the Rapid Object Recognition Training algorithms	38
3.1.1	Adaboost: a simple modification for convergence and speed	38
3.1.2	Building Classifiers with a set of Convex Hulls	42
3.1.3	Marking the positive examples	43
3.1.4	Accuracy and Performance Measurements	44
3.2	Preliminary Experiments	45
3.3	Building Beowulf Clusters	46
3.3.1	A brief description of the clusters	49
3.4	Performance of Clusters	51
3.4.1	Linpack performance results	51
3.4.2	An analysis of performance using mpptests	52
3.4.3	Ring test using jumpshot	54
3.5	Summary	56
4	Partial Occlusions in Face Detection	57
4.1	Introduction	57
4.2	The Training Process	58
4.3	Testing the classifiers	60
4.4	Experimental results and discussion	61
4.5	Summary	63
5	Hand Detection	65
5.1	Related work	65
5.2	Experiments with Haar-like features to recognise human hands	67
5.2.1	Preparing the Positive Set	67
5.2.2	Training	68
5.2.3	Detection	69
5.3	Results and Discussion	70
5.3.1	Accuracy	70
5.3.2	Performance	71
5.4	Summary	73
6	Rotational-invariant Approach	77
6.1	Background	78
6.2	Error sources in the computation of Haar-like features	80
6.2.1	Computing Haar-like features at 45°	81
6.2.2	Computing Haar-like features at 26.5° and 63.5°	82
6.3	Converting features to generic angles	84

6.3.1	Error Analysis for Pair of Equivalent Features (PEF)	85
6.4	Experimental Results and Discussion	86
6.4.1	Experiment 1: error analysis for PEFs using 0° and 45°	86
6.4.2	Experiment 2: error analysis for PEFs using 0° and 26.5°	89
6.4.3	Experiment 3: using PEFs to compute features in classifiers	93
6.4.4	Experiment 4: converting the original face classifier using tilted features	96
6.5	Summary	98
7	Real-time Moment Invariants	101
7.1	Related work	102
7.2	Geometric Moment Invariants	103
7.2.1	Hu's equations	104
7.2.2	Computing Moment Invariants from SATs	107
7.2.3	Preliminary Experiments	109
7.2.4	Limitations	111
7.2.5	Lighting Contrast Stretching	111
7.3	Concentric Discs	114
7.3.1	Circular Area (Discs)	115
7.3.2	Concentric Discs computation	116
7.4	Face Detection	122
7.4.1	Estimating angles using Hu's moments	122
7.4.2	Classifiers for Face Detection using Moments	123
7.5	Summary	126
8	Digits Recognition and Invariant Features	127
8.1	Related Work	128
8.2	Feature Extraction	130
8.2.1	The Concentric Discs Method (CDMI)	130
8.2.2	An Extended Feature Set for Normalised Central Moments	130
8.3	Training Methods	131
8.3.1	Convex Hull Classifiers	131
8.3.2	AdaBoost	132
8.3.3	Confidence Value for Multiple Classifiers	132
8.4	Experimental results	133
8.4.1	experiment 1: four types of classifiers	134
8.4.2	experiment 2: Ada-88- ψ with low training error	135
8.4.3	Experiment 3: η features with AdaBoost	137
8.5	Discussion	139

8.6	Summary	144
9	A Model for Parallel Classifiers	145
9.1	A Generic Platform for a Mobile Cluster	146
9.2	A Performance Model	146
9.2.1	Single Cascade	146
9.2.2	Concurrent Cascades	149
9.2.3	Communication	153
9.3	Implementing a mobile cluster using mini-ITX boards	154
9.3.1	Speedup	156
9.4	Summary	157
10	Final Conclusions	159
10.1	Summary	159
10.2	Limitations	160
10.3	Future Work	161
	Appendices	165
A	Implementation of SATs at 26.5°	165
B	Moments with SATs	171
B.1	Deriving The Equations For μ_{pq} From SATs (m_{pq})	171
B.2	4 th Order Moment Invariants Derivation	174
C	Additional Results for Digits Recognition	179
C.1	Experiment 1: additional Tables	179
C.2	Experiment 2: additional Tables	181
C.3	Experiment 3: additional Tables	182
	Bibliography	184
	Glossary	194

List of Figures

2.1	A generic computer vision Model	9
2.2	Histograms as features for computer vision systems	14
2.3	Examples of Haar-like Features.	15
2.4	The Summed-Area Table	17
2.5	Examples of tilted Haar-like Features.	18
2.6	AdaBoost applied in a colour classification problem.	20
2.7	A cascade of classifiers	23
2.8	Data decomposition	30
2.9	Four schemes for Parallel Computer Vision.	34
3.1	Classification using thresholds does not converge.	38
3.2	This set converges if the the classifier has four stages.	40
3.3	Example of convex hull used as a classifier.	43
3.4	Marking a face from the CMU-MIT database	44
3.5	ROC curve BASIC versus ALL features, 100 positive examples	46
3.6	ROC curve BASIC versus ALL features, 250 positive examples	47
3.7	ROC curve BASIC versus ALL features, 500 positive examples	47
3.8	ROC curve BASIC versus ALL features, 750 positive examples	48
3.9	ROC curve BASIC versus ALL features, 1000 positive examples	48
3.10	The Helix Beowulf-type cluster's topology	50
3.11	Linpack Rmax rating versus number of processors.	51
3.12	Point to point messages in Helix (short, blocking)	52
3.13	Point to point messages in Sisters (short, blocking)	53
3.14	Point to point messages in Helix (long, blocking)	53
3.15	Point to point messages in Sisters (long, blocking)	54
3.16	Goptest (Broadcast operation) running in 128 processors.	55
3.17	A ring test using 127KB message and 128 processors	55
4.1	Examples of partial occluded faces found in the three sets.	58
4.2	Example of a false positive object within the negative set.	59

4.3	The occlusion adding positive examples	60
4.4	Hits in several occluded faces.	62
4.5	ROC curve comparing Classifiers 2 and 3.	63
5.1	The basic hand images used on the training process.	67
5.2	Creating the positive set.	68
5.3	Using several cascades to recognise and determine rotation of hands	69
5.4	The Haar-like features chosen for the first step of each cascade.	71
5.5	ROC curves comparing the classifiers trained at different angles for test set A.	73
5.6	Examples of detection at different angles (from test set A).	74
5.7	Examples of images in test set B	75
5.8	Rates for cascades detecting hands in various angles	75
6.1	Normal features conversion to other types	80
6.2	Comparison between a normal and a tilted feature	81
6.3	Comparison between a 0° , a 26.5° and a 63.5° Haar-like feature.	82
6.4	Computing the SAT for 26.56° recursively	83
6.5	Corrections for SATs at 26.5°	84
6.6	Pair of Equivalent Features (PEF)	85
6.7	An example of PEF	86
6.8	Case where the value of the 22.5° feature should be MAX.	87
6.9	Case where the value of the 22.5° feature should be zero.	87
6.10	Scanning images for PEF comparison	88
6.11	Measuring errors for PEFs	88
6.12	First frame of Akiyo sequence and the chessboard images.	88
6.13	Maximum error vs angle for Akiyo (normal+ 45° PEFs).	89
6.14	Average error vs angle for Akiyo (normal+ 45° PEFs).	90
6.15	Maximum error vs angle for Chessboard (normal+ 45° PEFs).	90
6.16	Average error vs angle for Chessboard (normal+ 45° PEFs).	91
6.17	Maximum error vs angle for Akiyo (normal+ 26.5° PEFs).	91
6.18	Maximum error vs angle for Chessboard (normal+ 26.5° PEFs).	92
6.19	Average error vs angle for Akiyo (normal+ 26.5° PEFs).	92
6.20	Average error vs angle for Chessboard (normal+ 26.5° PEFs).	93
6.21	Hit ratio vs angle using 16 layers.	94
6.22	Hit ratio vs angle, kernel 36×36	94
6.23	False detection rate vs angle, kernel 36×36	95
6.24	The classifier 36×36 with various correction factors.	96
6.25	The hit rate for classifier 36×36 with limited range of features.	97

6.26	Detecting rotated faces using PEFs	97
6.27	Converting Haar-like feature maintaining the proportional area	98
6.28	Comparative ROC curve for upright and tilted cascades for the CMU-MIT database.	99
6.29	Some results for the tilted cascade converted from OpenCV.	99
7.1	Method overview	103
7.2	Grey scale samples of digits 9.	110
7.3	A sample of the working algorithm	111
7.4	Merging positive hits with Moments Invariants	112
7.5	An example of contrast stretching	113
7.6	Computing the moments for an approximation of a circular area	117
7.7	The Concentric Discs Moment Invariants approach (CDMI)	117
7.8	Test image from Gonzalez and Woods (2002).	118
7.9	Concentric disc area approximation	121
7.10	Angles estimated for faces	123
7.11	Face detection using CDMI.	124
7.12	The false detection rate versus the number of weak classifiers (1000 FERET samples)	125
7.13	The false detection rate versus the number of weak classifiers (2000 FERET samples)	126
8.1	The CMDI for handwritten digit recognition	130
8.2	Computing 12 η s per square area.	131
8.3	AdaBoost trained with sub-sets of the negative set	132
8.4	ROC curves for each classifier group, Ada-88- ψ -N.	137
8.5	ROC curves for each classifier group, η -84-N.	139
8.6	Number of errors caused by wrong choices between classifiers η -84-N.	140
8.7	Compared misclassifications for the three η and ψ feature types.	141
8.8	Distribution of the feature type usage per class for Ada-88- ψ -N.	142
8.9	Distribution of the feature type usage per class for η -84-N.	142
8.10	Examples of misclassifications for η with 84 features	143
8.11	Intersection of two competing classifiers for an unknown character.	143
8.12	Hypothetical handwritten digits showing a gradual transition between certain digits.	144
9.1	The proposed hardware platform.	147
9.2	Percentage of sub-windows per layer.	148
9.3	Features per layer for $Sub = 1000000$, $f_0 = 3$ and $C = 10$	149
9.4	% of the features computed up to layer x	150

9.5	The detector using multiple cascades in one slave node.	150
9.6	Time distribution for concurrent cascades running on one processor.	151
9.7	Frame rate for concurrent classifiers running on one processor.	151
9.8	Percentage of the total weak classifiers computed up to layer n.	152
9.9	Frame rate for concurrent classifiers running on one processor.	152
9.10	Time to broadcast a frame.	154
9.11	A mobile cluster.	155
9.12	Parallelisation scenario: runtime and communication times.	156
9.13	Speedups with different number of cascades and processors	158
A.1	SAT for 26.5°	165

List of Tables

4.1	The training parameters for the classifiers.	59
4.2	Testing images: number of occluded frames and non-occluded frames. . .	61
4.3	Hit rates (%) for the sets of images using different classifiers.	61
5.1	The kernel sizes for each cascade.	69
5.2	Hit ratios and false positives for the first test set (test set A).	72
5.3	Comparing the cascades for different angles using a rotated test set (test set B).	72
6.1	Performance of tilted cascades.	98
7.1	Runtimes for moment invariants computation.	110
7.2	Variances for contrast stretching.	114
7.3	11 moments computed over a square area for image in figure 7.8.	119
7.4	11 moments computed over an approximation of a circular area.	120
7.5	Variances for the approximation of concentric disc features from 1 to 0.5 in diameter.	120
7.6	Variances for the actual concentric disc areas.	121
8.1	Results reported on the MNIST database.	129
8.2	Training errors for four feature sets.	135
8.3	Overall Confusion Matrix for Ada-88- ψ -N.	136
8.4	Test errors for feature type ψ_n (90 concurrent classifiers) for the 10000 digits MNIST test set.	136
8.5	Overall Confusion Matrix for η with 84 features (η -84-N).	138
8.6	Test errors for feature type η_{pq} (90 concurrent classifiers) for the 10000 digits MNIST test set.	138
9.1	Results for the broadcasting times and SAT runtimes.	155
C.1	CH-11- ψ : false detection rate (%) per classifier.	179
C.2	CH-88- ψ : false detection rate (%) per classifier.	180

C.3	Overall Confusion Matrix for Ada-11- ψ -50.	180
C.4	Overall Confusion Matrix for Ada-88- ψ -50.	180
C.5	ψ -Ada-88-N: hit rate (%) per classifier.	181
C.6	ψ -Ada-88-N: false detection rate (%).	181
C.7	$\eta - 84 - N$: hit rate (%) per classifier.	182
C.8	$\eta - 84 - N$: false detection rate (%).	182
C.9	Number of errors caused by wrong choices between classifiers for η feature type, 84 features, N stages	183

List of Notations

A_{disc}	Sum of the pixels within a disc	116
A_{square}	Sum of the pixels within a square or rectangular area ..	115
α_t	Factor associated to the error of h_t	19
β_t	Factor associated to the error of h_t	20
C_n	Positional factor for moment invariants	113
$D_t(i)$	Weight of sample i at round t	19
η_{pq}	Normalised central moment	104
ε	Efficiency	153
f	Scaling factor for a kernel	25
$f(x)$	number of features computed up to layer x	148
f_s	Serial fraction of an algorithm	31
F	False detection rate	148
γ	Exponential factor for normalised central moments	104
H	Height of an image	24
$h_t(x_i)$	Weak classifier function	19
$i(x, y)$	Image pixel at (x, y)	16
$\bar{i}(x, y)$	Image resulting from a contrast stretching operation	26
$I(x, y)$	Summed-area Table element at (x, y)	17
$I_r(x, y)$	Rotated SAT element	18
m_{pq}	2D geometric moment of order pq	104
$m_{pq}(x, y)$	SAT element for order pq	107
\bar{m}_{pq}	Moment invariant with contrast stretching	113
μ	Mean (statistics)	26
μ_{pq}	Central moment	104
M	Width of a kernel	24
N	Height of a kernel	24
pt_n	Element of a SAT	115
$P(A B)$	Conditional probability of A, given B	133
ϕ_n	Hu's moment invariants	105
ψ_n	Flesser's moment invariants	107

s	Scale factor	16
Sub	Number of sub-windows in an image given s and t	147
$S(x)$	Number of sub-windows assessed by x layers of a cascade	148
σ	Variance (statistics)	26
t	Translation factor	24
T_m	Runtime on a multiple processor system	31
T_s	Runtime on a single processor system	31
θ	Direction of an object given by 2_{nd} order moments	122
V	Feature value	16
V_{normal}, V_{tilted}	Haar-like feature values	84
w_n	Constants that define a Haar-like feature	16
W	Width of an image	24
x_i	Feature array	19
\bar{x}	Ratio between first p order and zeroth order moments .	104
y_i	Class array	19
\bar{y}	Ratio between first q order and zeroth order moments ..	104

Abstract

This thesis studies rapid object detection, focusing on feature-based methods. Firstly, modifications of training and detection of the Viola-Jones method are made to improve performance and overcome some of the current limitations such as rotation, occlusion and articulation. New classifiers produced by training and by converting existing classifiers are tested in face detection and hand detection.

Secondly, the nature of invariant features in terms of the computational complexity, discrimination power and invariance to rotation and scaling are discussed. A new feature extraction method called Concentric Discs Moment Invariants (CDMI) is developed based on moment invariants and summed-area tables. The dimensionality of this set of features can be increased by using additional concentric discs, rather than using higher order moments. The CDMI set has useful properties, such as speed, rotation invariance, scaling invariance, and rapid contrast stretching can be easily implemented. The results of experiments with face detection shows a clear improvement in accuracy and performance of the CDMI method compared to the standard moment invariants method. Both the CDMI and its variant, using central moments from concentric squares, are used to assess the strength of the method applied to hand-written digits recognition.

Finally, the parallelisation of the detection algorithm is discussed. A new model for the specific case of the Viola-Jones method is proposed and tested experimentally. This model takes advantage of the structure of classifiers and of the multi-resolution approach associated with the detection method. The model shows that high speedups can be achieved by broadcasting frames and carrying out the computation of one or more cascades in each node.

Acknowledgements

I would like to thank my supervisors Dr. Martin Johnson and Dr. Chris Messom for their support and for providing an interesting and motivating environment for research.

I have also to thank my PhD colleagues who in the last few years shared the challenging activities related to research, in particular Farhad Dadgostar (with whom I share the authorship of two papers), and Chao Fan. Other colleagues in the institute that also have been actively discussing research issues and sharing ideas include Samuel Alexander and Tony Meyer. Thanks to Ravi Chemudugunta (with whom I share the authorship of one paper) for his collaboration in the mini-cluster project and for his UDP protocol for broadcasting frames.

This research would not be possible without the full support of the institute's head of department. I would like to thank Prof. Robert Mckibbin and Massey University for the opportunity to carry out the PhD project while working at IIMS. I also would like to thank the support and encouragement from Dr. Napoleon Reyes, Dr. Houssein Sarrafzadeh, Prof. Ken Hawick, Dr. Heath James, Dr. Chris Scoggings and all my colleagues at IIMS.

Thanks to my wife and my three children. The finalisation of my thesis was only possible with their full support and understanding.

Chapter 1

Introduction

1.1 Motivation

Computer Vision is a fascinating topic. Despite decades of research and despite all the improvements made both in hardware and in algorithms, one can appreciate that we are far away from matching the accuracy and the performance of mammalian vision. Even if, for many industrial applications, it is not necessary to have such a sophisticated approach, new applications arise where the current state-of-the-art struggles to meet a good response. For example, video compression could rely on the correct object segmentation to improve the performance and the compression ratio. However, there is no method that can reliably track objects such as cars or pedestrians in an arbitrary environment, where the lighting and the background can vary considerably.

In 2001, Viola and Jones (2001*a*) published an influential paper. For the first time, a simple approach enabled detection and tracking of human faces in real-time. Theoretically, their methodology would be applicable to any other object, however this was not the case. For many objects it is not possible to train an accurate classifier. To explain the limitations of the method it is necessary to study it in depth. It is useful to identify the limitations of the method and introduce modifications that could improve and generalise it.

The main objective of this thesis is to improve the knowledge of real-time object detection using simple feature extraction and learning methods. At the time of the writing of this document the most reliable and efficient method to detect faces is the Viola-Jones method (Viola and Jones, 2001*a*).

A list of objectives addressed in this thesis follows:

- improve detection under shadow and partial occlusions.
- deal with in-plane rotation of the detectable objects more efficiently.
- experiment with detection using other feature extraction methods.
- apply parallelisation strategies to the detection.

The main research questions are:

- Is it possible to improve the training performance by using a different set of features that can cope with rotation as well as scaling? Haar-like features have been used successfully, however they are not invariant to rotation (Viola and Jones, 2004). Attempts have been made to create systems in which rotation is dealt with, but a more generic feature extraction method without the need to retrain classifiers would be useful.
- Are moment invariants discriminative enough for generic detection and recognition tasks? It has been suggested that invariant features tend to be less discriminative (Postma et al., 1997). Also the number of independent moments that can be applied are limited by accuracy and noise issues (Teh and Chin, 1988).

1.2 Research Scope

The scope of this work is limited to feature-based systems. The training algorithms are based on modifications of the AdaBoost (Freund and Schapire, 1996), using simple thresholding to create weak classifiers. Although the approaches used in the experiments can be very fast, it is not among the objectives to optimise code. Even without optimisation, the new feature extraction method developed for this thesis achieved reasonable frame rates, when detecting objects in images acquired from a web camera (using a resolution of 640x480 pixels).

Training is carried out over limited databases of face images, hand images and handwritten digit images. The training is designed specifically to cover certain topics such as partial occlusion for face detection or a single gesture detection, rather than general classifiers.

1.3 Original Contributions

The main original contribution of this thesis is the development of a rapid feature extraction method based on moments that keeps invariance characteristics and can be used in a broad category of real-time applications. The method, called Concentric Disc Moment Invariants (CDMI), consists in extracting moment invariants from concentric discs of chosen sub-windows of the image. The increased number of dimensions in the feature set facilitates the training process, without the addition of noise.

Other contributions include:

- The proposal and evaluation of rotation invariant approaches for Haar-like features.

- The generation of special positive examples to train classifiers for partial occlusions and for hand detection.
- The proposal of a model for the use of parallel classifiers based on the Viola and Jones method. Characteristics of the parallelisation are evaluated using Beowulf clusters.

1.4 Publications related to this work

This research has resulted in the publications listed below.

- Barczak, A. L. C., Messom, C. H. and Johnson, M. J. (2003), Performance Characteristics of a Cost-effective Medium-sized Beowulf Cluster Supercomputer, *in* ‘LNCS 2660’, Springer-Verlag, pp. 1050-1059. This paper describes the performance of a Beowulf computer built by the Computer Sciences department at Massey University. The discussion and results are presented in section 3.4.
- Barczak, A. L. C. (2004a), Evaluation of a Boosted Cascade of Haar-like Features in the Presence of Partial Occlusions and Shadows for Real-time Face Detection, *in* Proceedings of the PRICAI 2004, LNAI3157’, Springer-Verlag, Auckland, NZ, pp.969-970. This paper is a short version of chapter 4.
- Barczak, A. L. C., Dadgostar, F. and Johnson, M. J. (2005), Real-time Hand Tracking Using the Viola and Jones Method, *in* ‘SIP 2005’, Honolulu, HI, pp. 336-341 and Barczak, A. L. C., Dadgostar, F and Messom, C. H. (2005), Real-time Hand Tracking Based on Non-invariant Features, *in* ‘IMTC2005’, Ottawa, Canada, pp. 2192-2199. These papers were published in preparation for chapter 5.
- Barczak, A. L. C. (2005), Toward an Efficient Implementation of a Rotation Invariant Detector using Haar-like Features, *in* ‘IVCNZ05’, Dunedin, NZ, pp. 31-36 and Barczak, A. L. C., Johnson, M. J. and Messom, C. H. (2006), Real-time Computation of Haar-like Features at Generic Angles for Detection Algorithms, ‘Research Letters in the Information and Mathematical Sciences’, v. 9, pp. 98-111. These papers were published as part of the preparation for chapter 6.
- Barczak, A. L. C. and Johnson, M. J. (2006), A New Rapid Feature Extraction Method for Computer Vision Based on Moments, *in* ‘International Conference in Image and Vision Computing NZ (IVCNZ 2006)’, Auckland, NZ, pp.395-400. This paper presents part of the moment invariants feature extraction presented in chapter 7.
- Barczak, A. L. C., Johnson, M. J. and Messom, C. H.(2005), A Mobile Parallel Platform for Real-time Object Recognition, *in* ‘ENZCon05’, Auckland, NZ, pp.

153-158. This paper relates to the work described in chapter 9. An extension of the model applied to a 4-nodes mobile Beowulf cluster is described in: Barczak, A. L. C. and Chemudugunta, R., Experiments with a Mobile Cluster for Real-time Object Detection, *in* '3rd International Conference on Autonomous Robots and Agents (ICARA 2006)', Palmerston North, NZ, pp. 303-308.

Additional co-authored papers that uses parts of the work presented in this thesis include:

- Dadgostar, F., Barczak, A.L.C. and Sarrafzadeh, A. (2005), "A Colour Hand Gesture Database for Evaluating and Improving Algorithms on Hand Gesture and Posture Recognition", *Research Letters in the Information and Mathematical Sciences*, Vol. 5, pp. 127-134.
- Messom, C.H. and Barczak, A.L.C. (2006), Fast and Efficient Rotated Haar-like Features using Rotated Integral Images, *in* 'proc. of the Australasian Conference on Robotics and Automation (ACRA2006)', Auckland, NZ,
- Reyes, N., Barczak, A.L.C. and Messom, C.H. (2006), Fast Colour Classification for Real-time Colour Object Identification: AdaBoost Training of Classifiers, *in* 'proc. of the 3rd International Conference on Autonomous Robots and Agents (ICARA2006)', Palmerston North, NZ, pp. 611-616.

1.5 Outline of the Thesis

Chapter 2 presents a literature review, focusing on feature based computer vision systems. It includes a discussion of the problems faced by detection algorithms in the presence of partial occlusion and rotation.

Chapter 3 describes the computational infra-structure used for the experiments carried out. This chapter also describes the methodology used to generate image samples and to collect data. The problems of AdaBoost convergence are discussed and a solution is proposed for finite sets of samples.

Chapter 4 describes the approach tested to overcome the problem suffered by the Viola-Jones method regarding partial occlusions. The experiments showed that by including random patches over the original positive set improves the response of the classifiers when faced with partial occlusions for face detection.

Chapter 5 shows experiments carried out to generically detect hands. Hand detection without the use of skin colour segmentation is a very difficult problem. Due to the different patterns generated by images that present hands at an angle and with different articulations, specialised classifiers are needed. An alternative method to generate samples is used in order to provide classifiers able to cope with rotation.

Chapter 6 extends the concept of rotation by proposing alternatives for Haar-like feature extraction. A variation of a summed-area table to extract Haar-like features at an arbitrary angle is proposed. A detailed analysis of the errors is presented.

Chapter 7 proposes a new feature extraction method based on moment invariants. The method is capable of increasing the dimensionality of the moment invariants set by using concentric circular areas of extraction (CDMI), and it keeps the rotation and scaling invariance. The performance and accuracy are experimentally tested and discussed. A simple method to extract moment invariants under different lighting conditions is also proposed. Face detection is investigated by training classifiers with AdaBoost.

Chapter 8 discusses the application of the moment invariant method developed in chapter 7 applied to hand-written digit recognition. Experiments are carried out using the MNIST database. Classifiers are trained, using both rotation invariant (CDMI) and non-invariant moments (Central Moments extracted from concentric squares). Although the technique could not be used on its own, the results are promising.

Chapter 9 proposes a model for parallel cascade classifiers, which was drawn based on the performance characteristics of clusters obtained from previous experiments. It is possible to estimate the expected frame rate for a parallel detection system using this model.

Chapter 10 presents the final conclusions and recommendations for future work.

Chapter 2

Literature Review

This chapter presents a review of the literature focusing on two topics: rapid detection (to some extent related to recognition and tracking) and parallel computing applied to computer vision. In section 2.1, image-based object detection, object recognition, and object tracking methods are discussed. In section 2.2, the Viola-Jones method (Viola and Jones, 2001*a*) is discussed in greater detail, including discussions about the current limitations of the method. In section 2.3, the parallel approach using a cluster of computers is described and its implications for machine vision are discussed. Finally, in section 2.4, a summary of the chapter shows the limitations of the Viola-Jones method, its relationship with other methods and the direction for the experiments carried out for this thesis.

2.1 Computer Vision and Rapid Object Detection

A generic computer vision system is usually presented as a set of serial steps. Figure 2.1 shows a generic model, which is based on Computer Vision models presented by Luo (1998) and Awcock and Thomas (1996) and DeRidder (2001). Each step has a specific function in the computer vision chain. Between each step the input data differs from the output data in size and nature. There are no general rules that guarantee that the data size decreases when passing through one step, although that is what usually happens with feature-based systems. There are other systems that increase dimensionality in intermediate steps. Examples of the latter systems are Neural Network-based system such as LeCun et al. (1998) and Rowley (1999). The final steps of figure 2.1 have limited inputs, as the main task of a computer vision system is to analyse and act upon the information obtained via the processing of images. Typical responses are the identification of certain events (e.g. some movement in front of the camera) or the return of the position where an object is to be found.

The various steps presented in figure 2.1 often overlap, so it is not always possible to isolate them completely or present them separately on real-life computer systems. For the

purpose of the discussions in this thesis, computer vision systems were classified into two categories, depending on the way features are extracted and interpreted:

- **feature-based systems:** the feature extraction methods are clearly separable from the other steps. The interpretation of the values obtained on the feature extraction step can be implemented independently. Example: Viola-Jones method (Viola and Jones, 2001*a*)
- **mixed-stage systems:** in these systems the feature extraction method either depends on training or is inherently mixed with other steps. Example: Neural Network-based computer vision systems such as LeCun et al. (1998) and Rowley (1999).

Feature-based systems are easier to analyse because one can get a set of numbers from the feature extraction step. Also training, detection, and feature extraction can be completely separated. Mixed-stage systems, on the other hand, work like a “black box”: the feature extraction process is somewhat hidden and mixed with the recognition step. Mixed-stage systems also have other problems. For example, de Sa (2001) points out that one disadvantage of Neural Networks is that no semantic information is available. While in statistical classifiers, one can often describe how the output was arrived at, “such a perception is usually impossible in the case of Neural Networks”.

Based on the colour characteristics, one can also classify object detection systems as:

- Colour-based: a colour segmentation is an essential step in the computer vision chain.
- Shape-based: grey-scale images are used so that the feature extraction step has all the information necessary.

For feature-based systems, one can classify object detection based on the the feature’s mathematical nature in terms of their ability to cope with transformations:

- Invariant features: there is no need to recompute features even if the images are transformed.
- Non-invariant features: the feature values are dependent on the transformations.

2.1.1 Real-time Object Detection, Recognition and Tracking

The area of Object Detection and Recognition has made significant progress in the last few years. Many algorithms developed recently in this area relate to human face detection and recognition due to its potential applications in security and surveillance. Yet, a generic, reliable, and fast human face detection was, until very recently, impossible to achieve in real-time.

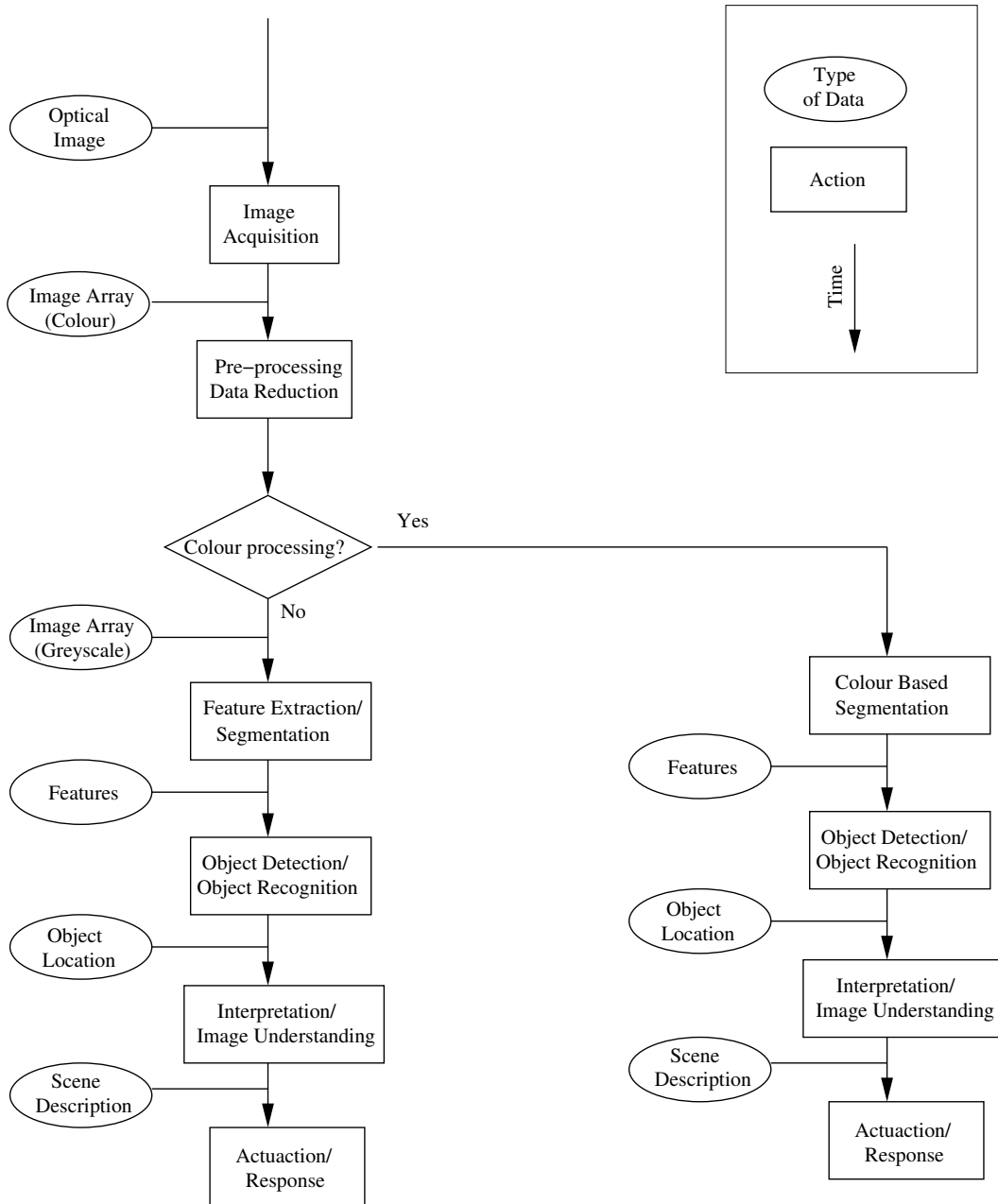


Figure 2.1: A generic computer vision Model (adapted from Awcock and Thomas (1996)).

The concepts involved in object detection, object recognition, and object tracking often overlap. Each of these computer vision techniques try to achieve the following:

- Object Tracking: dynamically locates objects by determining their position in each frame.
- Object Detection: locate generic classes of objects in the image (such as faces).
- Object Recognition: classify specific objects in the image (such as a face that belongs to one individual, a certain printed character etc)

In practice, the same methods with minor variations are used to achieve one of the three tasks presented above. For example, by detecting objects in a video sequence in real-time, one can also achieve tracking. Other methods use a completely different approach, tracking with previously marked images without relying on detection or recognition. An example is seen in Lucas and Kanade (1981), which uses an optical flow approach to track previously marked objects. The robustness of the latter approach is weaker than using some form of detection, since optical flow related methods do not use pattern recognition techniques. Optical flow techniques can be very susceptible to mistakes if the camera is not stationary.

Many algorithms for object recognition are too slow to be considered for use in real-time applications; even with relatively fast feature calculations such as FFTs, wavelets or eigenvalues, it is still a challenge to produce a robust classifier that is fast enough to be used in real-time on a desktop computer.

Tracking specific objects such as frontal faces or cars is an important sub-set of the generic video detection problem. In these cases, geometric characteristics of the objects can influence the method that is being applied. Objects being tracked may be rigid, semi-rigid (such as hands articulating or a pedestrian walking) or completely flexible. The environment, notably lighting, also plays an important role because it can change the patterns related to the image. Rowley (1999) mentioned that the difficulty of detecting objects varies according to their nature: “The car detection problem has several sources of difficulty that are not present in the frontal face detector. In the face detector, the neural network had to deal with essentially one pattern, that of a frontal face, which does not vary much from person to person. For the car detector to work, the neural network would have to detect any car shape (of which there are a great variety), and to deal with appearance changes caused by specular reflections of the environment off of the car bodies.” Detecting cars from one specific view point could be useful for specific applications, but it still does not explain why generic detectors are so difficult to train. Human faces have a very specific geometric pattern that can be explored to create classifiers.

Accuracy and performance in computer vision can be measured in many different ways, and often the literature refers to “de facto” standards established over the years.

For object detection, accuracy is measured in terms of hit rates (percent of correctly identified objects) and false detection rates (percent of images that do not belong to that object class). However, there are many problems with this simple measurement. It is often difficult to tune systems to their optimum response, as improving the hit rate also increases the false detection rate. The ROC (receiver operating characteristic) curve can be used to compare different systems (Egan, 1975). Performance is usually associated with speed, most commonly a frame rate if the images are coming via cameras or video sequences.

2.1.2 Face Detection

Gong et al. (2000) published a complete review of different methods for face detection, including discussions on commercial applications and databases used for benchmarking.

A survey by Yang et al. (2002) defines Face Detection as the location of faces in an arbitrary image. They found that all published methods could fit into four main categories (that overlap each other): knowledge-based methods, feature invariant approaches, template matching methods, and appearance-based methods. More importantly, there are methods that rely on skin colour segmentation as a crucial step. Some methods can be generalised to find other objects, while others are inherently limited to face detection tasks.

Sung and Poggio (1998) developed one of the earliest methods for face detection based on Neural Networks. They carried out experiments with several distance measurements for the training and different Neural Networks set ups. The performance and accuracy were measured using their own data set, which was later incorporated into the CMU-MIT data set (Rowley et al., 1998*b*).

Schneiderman and Kanade (2000) (see also Schneiderman and Kanade (1998)) developed one of the earlier successful object detection methods using simple decision rules and wavelets. The system was reported to get about 85% to 95% detections with a relatively high number of false detections, but was quite slow (about 5 seconds to scan a 320x240 pixels frame).

Rowley et al. (1998*b*) developed a face detector using an image pyramid and extracting 20x20 pixel sub-windows. After correcting lighting and equalising the sub-window, the system passed the sub-window through a NN filter, where a face or non-face label was the result. Several experiments using different NN (or a combination of NN) yielded detection rates between 85% to 93%. Face detection was high in some cases, with the best tradeoff resulting in 85.4% detections. The system was later expanded to be rotational invariant (Rowley et al., 1998*a*).

An influential paper from Viola and Jones (2001*a*) received considerable attention from the Computer Vision community and it has generated many other experiments.

Viola and Jones (2001*a*) created a complete method for face detection based on Haar-like features. Using what they called Integral Images (originally called Summed-area Tables by Crow (1984)), a feature only needed a few lookups to be computed at any position and scale. Viola and Jones computed Haar-like features in such a way that the features were scaling invariant. Their approach was extended and was partially implemented in open source image processing libraries such as OpenCV (Bradski, 2002, 2000). Training was based on a variation of AdaBoost (Freund and Schapire, 1996) where the classifiers were organised in a cascade. More details about this method is discussed in the next section.

Many variations on Viola-Jones method have been published. Li et al. (2002) modified AdaBoost (calling it FloatBoost) and improved the accuracy of the multi-view detector for faces. The new boosting algorithm worked by eliminating some of the weak classifiers that did not improve the overall classification. Other proposed variations include Lienhart and Maydt (2002), Cristinacce and Cootes (2003), McCane and Novins (2003), Lienhart, Kuranov and Pisarevsky (2003), Menezes et al. (2004), and Mita et al. (2005).

Garcia and Delakis (2004) used a Convolutional Neural Network to detect faces. Convolution Networks (LeCun et al., 1998) were originally created for applications in OCR. This method achieved multi-view face detection, limited to $\pm 20^\circ$ in plane rotation and $\pm 60^\circ$ perpendicular rotation. The results were promising, although the system was slower and less accurate than Viola-Jones. The authors reported 0.25 seconds for a 384x288 pixels frame and about 90% hits on the CMU.

Osadchy et al. (2005) also used a Convolutional Neural Network as the basic architecture for a face detector. Their system used raw pixels to learn (via a special Neural Networks approach) both low-level features and high-level representation. The training integrated feature detection with the classifier. This approach is very promising for multi-view face detection. The idea was somewhat opposed to the prevalent one that classifiers should be trained separately for different tasks. Osadchy et al. (2005) claimed that by joining multi-view detection and pose estimation in one classifier, it generalised better and required fewer examples. However, this idea does not seem to work with the original Viola-Jones method, as it is known that joining tasks in a monolithic classifier yields too many false detections (Jones and Viola, 2003). ROC curves presenting the results actually show that frontal faces are better recognised. Rotated faces yielded relatively good results, but faces in profile are the worst (with an acceptable result of 75% for one false detection per image). The performance of their system compared to Jones and Viola (2003) was very similar. The number of examples was higher (52850 images of 32x32 pixels), although the training time was around 26 hours, very short in comparison to the Viola-Jones method. Detection achieved a rate of 5 frames per second, which is somewhat slower than what can be achieved with the Viola-Jones method.

2.1.3 Invariant Features and Non-invariant Features

In feature-based methods, the features used can be invariant or non-invariant. Invariant features do not depend on geometric transformations of the image. Usually, the transformations required are rotation, translation and scaling, but it can also include affine transformations, contrast stretching etc. Non-invariant features, on the other hand, are dependent on these transformations. Invariant features are usually invariant to one or more transformations such as translation, rotation, scaling, general affine transformations etc. A review of the general problem of invariant and feature-based object recognition is discussed by Wood (1996) and by Postma et al. (1997).

Feature extraction may yield feature sets that are unable to discriminate classes of images. For example, Postma et al. (1997) cited the “scrambling problem”. Two images can be modified in such a way that they have identical histograms. Features extracted from histograms do not keep information about the position of the pixels. Even though histograms might be useful for eliminating a small portion of all the possible false detections, their discriminating characteristics are insufficient to unambiguously classify images. Our early experiments with histograms are illustrated in Figure 2.2. The second image was modified to fit the histogram of the first image.

Marr (1982), in his classic book, proposed the use of features which go beyond a set of raw pixels and are partially based on a better understanding of the physiology of the human eye and human vision strategies. According to Marr, there was evidence that edge and corner detectors are used by human vision. However, generic detectors based only on these features did not yield reliable results, unless for very simple problems in controlled environments. It is not always clear how to correlate certain objects with edges and corners. It is also difficult to automatically tune parameters in order to get good edges (see for example Stenger (2004) using Canny edge detectors). Although systems using such approaches were developed (an example is described by Gong et al. (2000)), problems such as face detection were only solved satisfactorily using more sophisticated techniques and features that do not depend directly on edge detection.

Another aspect of the feature space is the dimensionality. Marr (1982) and Postma et al. (1997) discussed and presented some evidence that the brain uses a high dimensional feature space for images (in fact much higher than the image space itself in the retina). A typical problem in machine learning is the so-called “curse of dimensionality”. Usually, the feature space has to have a minimum number of dimensions in order to allow classification. But, if the number of dimensions is too high, the training process may be unfeasible or becomes impractically long.

For some time, the focus in Object Recognition was on training algorithms rather than features or the geometry of the objects. Successful work was done in character recognition, but trying to apply similar algorithms to any type of object was elusive for a long time.

In the 90's, researchers began to realise the importance of the features rather than the training. Ripley (1996), for example, stressed the fact that the choice of features can be more important than the training itself. "Much of the enhanced success of Zip code recognition systems has come from better features rather than through more complicated classifiers" (Ripley, 1996). Freund (1998) also observed that it was common that good features beat good training.

Optimum selection of features that require less computation time and are more accurate is still an unsolved problem. Research into feature-based object recognition, according to Postma et al. (1997), suffers from three, as yet, unsolved problems:

1. what makes a good feature: characteristics such as computational complexity, discrimination power, and robustness to noise.
2. how many features should be used: the "curse of dimensionality" describes the problem well, as the more features a system has, the more information about the image is obtained. However, the more features the harder the training process.
3. how to cope with insensitivity to spatial information: a typical example is the histogram, with which one can differentiate images, but with which all the spatial information is lost. In figure 2.2 there are two images with the same histogram, showing that histograms do not have good discrimination powers.

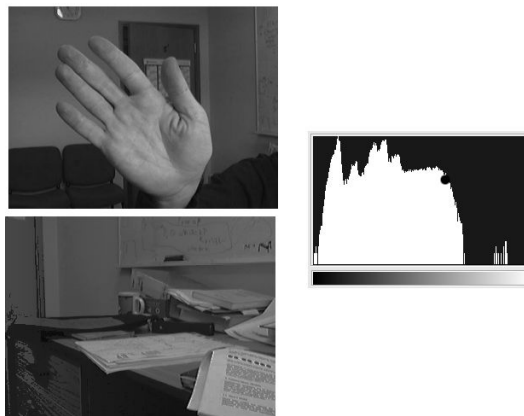


Figure 2.2: Histograms as features for computer vision systems: the hand and the table have identical histograms.

An important class of invariants are related to geometric moments (Hu, 1962). Chapters 7 and 8 of this thesis is dedicated to these features.

2.2 Rapid Object Detection: a review of the Viola and Jones Method

This sub-section examines some of the details about the Viola and Jones (2001*a*) method. Three characteristics of their method were important to obtain performance and accuracy. The first was the use of a special data structure that allows a very fast calculation of features. The second was the use of AdaBoost, a training algorithm first proposed by Freund and Schapire (1999) allowing the selection of the few best features that fit the positive example. The third was the combination of the resulting classifiers in a cascade, allowing a fast elimination of sub windows that do not contain the object. Viola and Jones work was, in part, based on work done previously by Rowley et al. (1998*a*) and by Papageorgiou et al. (1998).

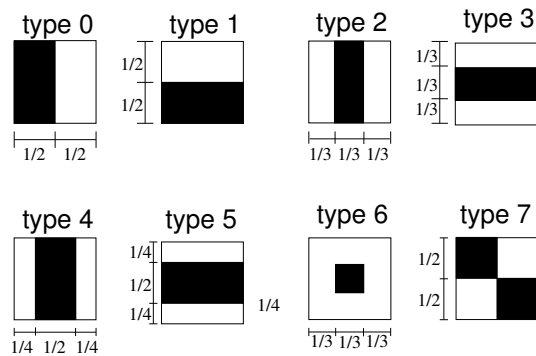


Figure 2.3: Examples of Haar-like Features.

The Viola-Jones method has been used mainly for face detection (Viola and Jones, 2001*b*), face recognition (Guo and Zhang, 2001; Froba et al., 2003), and hand detection (Kolsch and Turk, 2004*a,c*; Ong and Bowden, 2004). Other uses include robot-soccer ball detection (Mitri et al., 2004), license plate detection (Dlagnekov, 2004), and even for ecological applications, such as wild life surveillance (Burghardt, Thomas, Barham and Calic, 2004; Burghardt, Calic and Thomas, 2004).

2.2.1 Haar-like Features

The first contribution of Viola and Jones algorithm was the fast calculation of features. The features used by their method are called Haar-like features (figure 2.3) because of their similarity with the Haar Basis functions. The basic shape of a Haar-like feature used is rectangular. The sum of the pixels of a rectangular region was subtracted from the sum of the pixels of another region of the image.

The Haar like features are defined as the difference of the sum of the pixels of rectangular areas. In figure 2.3, feature types 0 and 1 are two rectangular Haar-like feature.

The sum of the pixels in the shaded rectangle is subtracted from the white rectangle. Feature types 2, 3, 4 and 5 are three rectangle features where sum of the pixels of the dark rectangle is subtracted from the sum of the pixels of the other two. Type 6 is a two rectangle feature where the dark area is a rectangle inside the main rectangle (not used in the original Viola and Jones (2001a)). Type 7 is a four rectangle feature and the difference between diagonal pairs of rectangles is the feature value.

The values of features in figure 2.3 are computed by:

$$V = \frac{w_1 \sum_{i \in \text{area1}} i(x, y) + w_2 \sum_{i \in \text{area2}} i(x, y) + w_3 \sum_{i \in \text{area3}} i(x, y)}{s^2} \quad (2.1)$$

Where w_i are constants inversely proportional to the area sizes (in number of pixels) and s is the scale factor, with $s \geq 1$. In practice, it is easier to compute an area correspondent to the entire feature with $w_1 = 1$ and subtract the darker areas separately. So $2w_2 = -w_1$ for types 0 and 1, $3w_2 = -w_1$ for types 2 and 3, $2w_2 = -w_1$ for types 4 and 5, $9w_2 = -w_1$ for type 6 and $2w_2 = 2w_3 = -w_1$ for type 7.

These features are said to over-represent the objects, as they are redundant and more numerous than the image intensities themselves. In order to achieve scaling invariance, it suffices to divide the feature values by the area (measure in pixels). These features are computed at various scales by using what Viola and Jones (2001a) called the *integral image*, also called Summed-area Tables (SAT), discussed in the next section.

Lienhart and Maydt (2002) worked on some improvements for better accuracy with Viola-Jones method by adding tilted Haar-like features (feature types 10 to 17 of figure 2.5). These tilted features represent diagonal areas of the image. Tilted Haar-like feature computation uses the same SAT data structure. A special tilted SAT must be created. They reported some improvement in the accuracy, but it required extra time for training because the feature set was enlarged. The real-time constraints for detection were maintained. Although it was not proved beyond doubt that these extra features were essential to improve accuracy, they could potentially be used to create tilted classifier for searching rotated objects. Further exploration of this idea is discussed in chapter 6.

2.2.2 Summed-area Tables or Integral Images

Integral images, formerly known as Summed-area Tables (SAT), can be traced back to a paper by Crow (1984). Originally, Crow used SATs for texture mapping. A SAT is defined as a matrix in which each element contains the sum of all the pixels that belong to upper left parts of the original image. Given an image $i(x_i, y_i)$, the Summed-area Table $I(x, y)$ is:

$$I(x, y) = \sum_{x \leq x_i, y \leq y_i} i(x_i, y_i) \quad (2.2)$$

where: $i(x_i, y_i)$ is the set of pixels of the image.

Once a Summed-area Table is created for a certain image, the sum of rectangular areas over the image can be computed with 4 lookups. Figure 2.4 shows an example. With four points of the Summed-area Table the sum of the pixels contained in the grey area is:

$$\sum pix = pt4 - pt3 - pt2 + pt1 \quad (2.3)$$

Haar-like feature values are rapidly computed at any position and any scale directly from the SATs. The Haar-like features in figure 2.3 need a certain number of lookups depending on their shape. For example, types 0 and 1 need 6 lookups each, types 2 to 6 need 8 lookups each and type 7 needs 9 lookups.

A recursive algorithm for creating the table can be based on the following equation:

$$I(x, y) = I(x - 1, y) + I(x, y - 1) - I(x - 1, y - 1) + i(x, y) \quad (2.4)$$

Where $I(x, y)$ is the SAT element and $i(x, y)$ is the image element for the point (x, y) .

In order to avoid negative indexes, the SAT is padded with zeros in the first row and column. The complexity of the SAT calculation is $O(N)$, where N is the total number of pixels in the image.

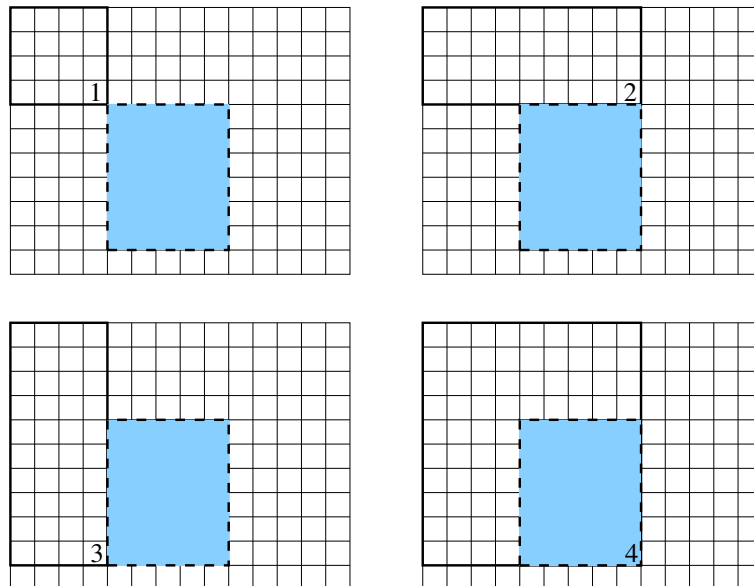


Figure 2.4: The SAT. The sum of the pixels in the shaded area is $area4 - area3 - area2 + area1$ ($area1$ was subtracted twice as it is contained in areas 2 and 3)

Lienhart and Maydt (2002) extended the Haar-like feature set including new tilted features (areas at 45° angle) (figure 2.5). To compute Rotated SATs I_r recursively, they

used the following approach:

$$I_r(x + 1, y + 1) = I_r(x, y) + I_r(x + 2, y) - I_r(x + 1, y - 1) + I(x, y) + I(x, y - 1) \quad (2.5)$$

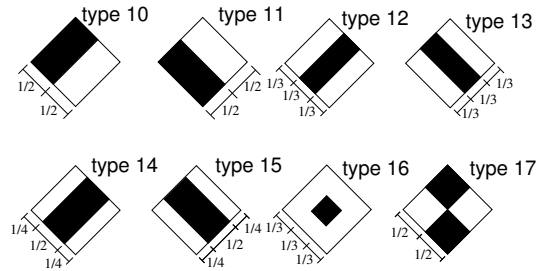


Figure 2.5: Examples of tilted Haar-like Features.

2.2.3 Adaboost

Viola and Jones have chosen a training algorithm called *AdaBoost*, first proposed by Freund and Schapire (1999). Generally speaking, *Boosting* is a statistical method that modifies the original distribution of positive and negative examples, and combine simple rules (called “weak” classifiers) to compose a stronger classifier. AdaBoost is most commonly used for binary classification, but with minor modifications it can also deal with many classes. The so-called “simple rules” could be any classification method that, for a given distribution, is just slightly better than chance. One could use, for example, simple Bayesian rules, simple histogram based threshold, decision trees etc. It has been argued that due to the difficulty of modeling distributions, a rule based approach can be more accurate and effective than a model based approach (Freund, 1998). AdaBoost uses the original distribution to compute rules that fit the data to a given hit rate and a false detection rate. The resulting classifier is a combination of these rules.

AdaBoost is considered one of the best learning algorithms introduced in the last decade (Hastie et al., 2001). Advantages of AdaBoost are:

- it is simple to implement;
- it generalises training for large dimensions;
- and it automatically selects features from large feature spaces.

In the last few years some disadvantages have been reported (Rudin et al., 2004):

- it does not necessarily converge and might go into a cycle.
- robustness against overfitting cannot be guaranteed.

- multiple classes are difficult to train.

The simplest and most popular version is the one proposed by Freund and Schapire (1996). Given a two class data set:

Algorithm 1 *AdaBoost*((*value, class*), *T*)

The input is a set of training examples $(x_1, y_1), (x_2, y_2), \dots, (x_n, y_n)$ where x_i is a feature $\in X$ and y_i is a class $\in \{-1, +1\}$, and a number of rounds T . It outputs a linear combination of weak classifiers $h_t(x_i)$, each a factor α_t .

Initialise weights for each element $D_1(i) = 1/n$ where n is the number of elements

For $t = 1, 2, \dots, T$:

1. Train a weak classifier h_t (e.g. a simple threshold) using the weights in D_t so that $h_t \in \{-1, +1\}$

2. Compute the error associated with the weak classifier:

$$error_t = \sum_i (D_t(i) h_t(x_i) y_i) \quad (\text{the sum of the elements' weight that are incorrectly classified})$$

3. Compute $\alpha_t = 0.5 \ln\left(\frac{1-error_t}{error_t}\right)$

4. Update the weights $D_{t+1}(i)$ such as D_{t+1} is a new distribution:

$$D_{t+1}(i) = D_t(i) \exp(-\alpha_t y_i h_t(x_i))$$

(the weights decrease if the element is classified correctly or increase if it is classified incorrectly)

$$\text{normalise the weights } D_{t+1}(i) = \frac{D_{t+1}(i)}{\sum_i D_{t+1}(i)}$$

5. After T rounds the final classifier is: $H(x) = \text{sign}(\sum_t \alpha_t h_t(x))$
-

This version of the algorithm works with one dimension. With a minor modification the algorithm is suited for a larger number of dimensions. For each round of AdaBoost, instead of computing one hypothesis h_t , one hypothesis for each dimension is found, and the one with the smallest error is chosen. Sets with a very large number of dimensions can be successfully trained using AdaBoost because it essentially does a greedy search (Freund and Schapire, 1999). Each hypothesis tries to find the correlation between the training set and a certain feature. After the re-weighting it is very likely that another feature (represented as one coordinate of the feature space) is chosen. The process goes on until the required training error is achieved.

To understand the mechanism behind AdaBoost, Figure 2.6 shows an example of colour classification in an earlier experiment for this work. A number of pixels were represented by their values in HSV¹ colour space, of which it is assumed that there was no correlation with V (so only two dimensions are used for training). Each round of AdaBoost increased the weights of incorrectly classified examples. Thresholds were found for each distribution so the error was minimised for both H and S dimensions. The best threshold was chosen,

¹H is hue, S is saturation and V is value.

and the re-weighting process was carried out once more. After 5 rounds, the final classifier divided the HxS space in 12 rectangles. Each rectangle only contained either positive or negative examples and therefore the training error was zero.

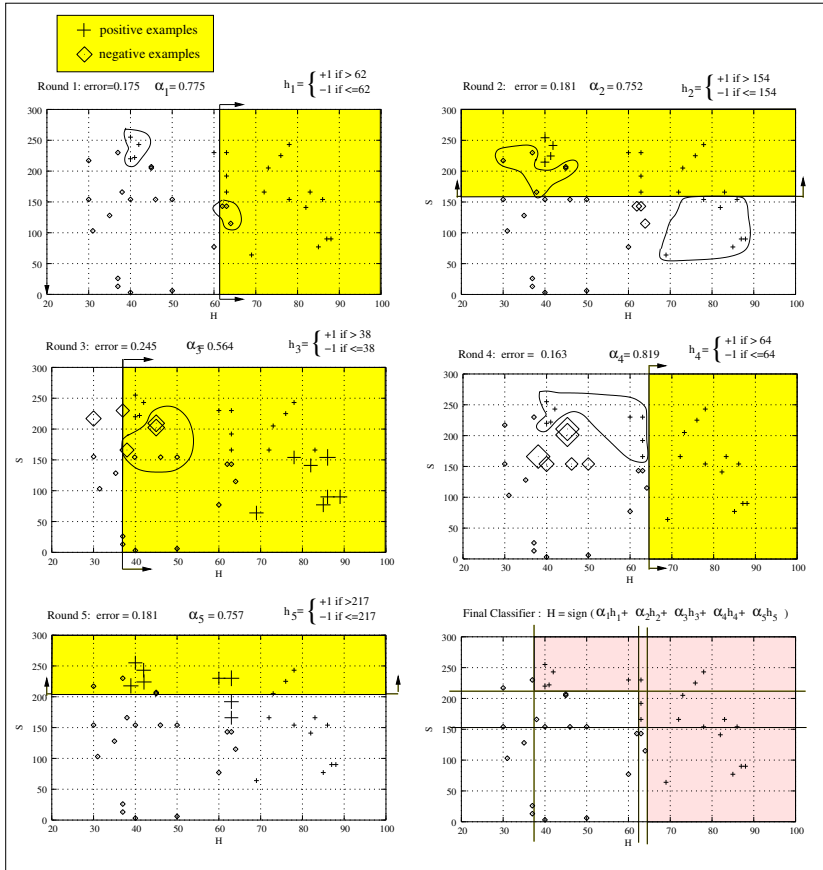


Figure 2.6: AdaBoost applied in a colour classification problem.

Viola and Jones (2001a) used algorithm 2, a modified version from the original AdaBoost. A single strong classifier did not suffice to achieve the real-time constraints of the detection application. They solved this problem using a cascade of classifiers. The version of AdaBoost used by Viola and Jones was also modified to accept the expected hit and false detection rates rather than fixing the number of rounds T .

Lienhart, Kuranov and Pisarevsky (2003) experimented with slightly different boosting methods, namely Discrete AdaBoost, Real AdaBoost, and Gentle AdaBoost. Discrete AdaBoost uses binary weak classifiers. Real AdaBoost uses a probability function as the weak classifiers. Gentle AdaBoost uses a regression function by weighted least-square of y_i to x_i . Lienhart, Kuranov and Pisarevsky (2003) results showed that gentle AdaBoost slightly outperformed the other two methods for the case of face detection.

AdaBoost is able to choose the most significant features given a few thousand positive examples and negative examples. Thus, even if the feature space has a very high dimen-

Algorithm 2 AdaBoost_ViolaJones $((value, class), P_{requ}, F_{requ})$

The input is a set of training examples $(x_1, y_1), (x_2, y_2), \dots, (x_n, y_n)$ where x_i is a feature $\in X$ and y_i is a class $\in \{0, 1\}$, and a number of rounds T . It outputs a linear combination of weak classifiers $h_t(x_i)$, each a factor α_t .

Initialise weights for each element $D_1(i) = 1/n$ where n is the number of elements. Initialise the positive hit rate $P = 0$, the false detection rate $F = 1$ and the round $t = 0$. The expected positive hit rate for this round is typically $P_{requ} > 0.99$ and the expected false detection rate for this round is typically $F_{requ} < 0.30$.

While $P < P_{requ}$ and $F > F_{requ}$:

1. Normalise the weights $D_{t+1}(i) = \frac{D_t(i)}{\sum_i D_t(i)}$ (so $D_{t+1}(i)$ is a distribution)
 2. For each feature j train a weak classifier $h_{t,j}$ (e.g. a simple threshold) using the weights in D_t so that $h_{t,j} \in \{0, 1\}$
 3. Compute the error associated with the weak classifier of each feature:

$$e_{t,j} = \sum_i (D_t(i) h_{t,j}(x_i) y_i)$$
 (sum the elements' weights that are incorrectly classified)
 4. Choose the classifier $h_t(x_i) = h_{t,j}$ with the smallest error $e_{t,j}$
 5. Compute $\beta_t = (\frac{e_t}{1-e_t})$ and $\alpha_t = \ln(1/\beta_t)$
 6. Update the weights $D_{t+1}(i)$:

$$D_{t+1}(i) = D_t(i) \beta_t^{1-h_t(x_i)}$$

(the weights decrease if the element is classified correctly or increase if it is classified incorrectly)
 7. After $t + 1$ rounds the classifier is: $H(x) = \begin{cases} 1 & \sum_t \alpha_t h_t(x) \geq \frac{1}{2} \sum_t \alpha_t \\ 0 & otherwise \end{cases}$
 8. Compute the positive hit rate P and the false detection rate F for round $t + 1$
-

sionality, it is still possible to train a classifier with good accuracy. However, there may be so many features to be examined that even using just a few thousand examples can make the training process a very long endeavour. It can take weeks of run-time to obtain good classifiers, a problem that Viola and Jones (2004) solved by simple parallelisation. Once the classifier is created, it can be used in real-time detection.

McCane and Novins (2003) discussed the problems of long training times and found a solution to speedup the training. Rather than rely on an exhaustive search for finding the best weak classifier, they used an heuristic optimisation procedure based on the error function. The solution showed an improvement of 300 times for specific cases.

Li (2005) proposed an alternative called FloatBoost. He demonstrated that, in many cases, the rules added to the final classifier do not necessarily improve its accuracy. Depending on the dataset, some of these rules could be safely deleted, improving the overall performance without any loss of accuracy.

Wu et al. (2003) argued that the way Viola and Jones used the AdaBoost algorithm is an indirect way of feature selection and other methods could train a cascade more efficiently. They used a forward feature selection approach and claimed to have achieved good classifiers with about a 100th of the time taken by Viola and Jones. However, it has not been proved that simple feature selection solves the problem in a better way. In fact, the evidence supports the opposite.

2.2.4 Cascades of Classifiers

The third contribution of Viola and Jones (2001*a*) is the arrangement of partial classifiers in a cascade. Each stage is only required to eliminate slightly more than 50% of false detection as long as it kept the positive hit rate close to 100%. Each stage examines what are the chances that a particular sub-window has in presenting the object being detected. After 20 or 30 stages most of the false positives are eliminated. The remaining sub-windows are eventually scanned by the last cascade. What remained should be the sub-windows that contained the image of the object for which the classifier was trained. Figure 2.7 shows an example of the expected results of cascading classifiers in this way.

As an example, lets assume that each stage of the cascade (each stage is itself a classifier) eliminates 50% of the false positives, while it wrongly eliminates about 0.2% of true positives. If there are 20 stages (Viola and Jones, 2001*a*):

$$0.50^{20} = 0.00000095 = 0.000095\% \text{ of false positives}$$

$$0.998^{20} = 0.961 \simeq 96\% \text{ hit rate}$$

Viola and Jones (2001*a*) claimed that similar accuracies could be achieved using a monolithic classifier with one layer (or stage), but the performance of a cascade (with multiple stages) was much better. In fact, most of the false sub-window are eliminated by the first few stages. The multi-stage classifier that Viola and Jones trained for face

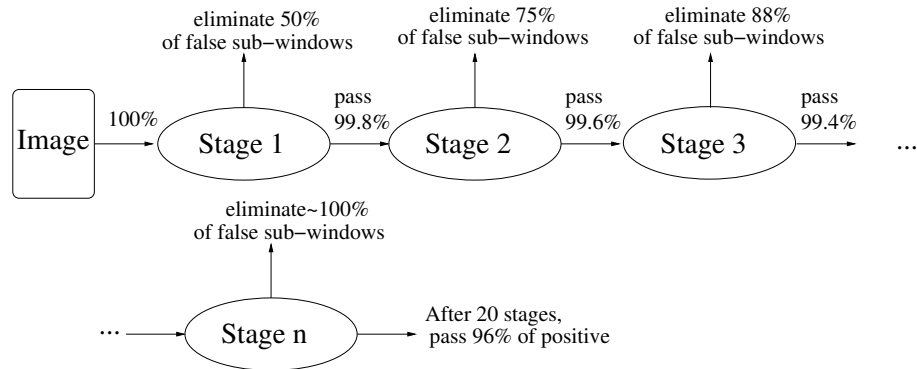


Figure 2.7: A cascade of classifiers (Adapted from Viola and Jones (2001a)).

detection has just a few weak classifiers in the first few stages. The last few layers have more features, as it takes more weak classifiers to eliminate false sub-windows that present a much more similar pattern when compared to face images.

McCane et al. (2005) discussed the optimisation of cascades classifiers. Rather than starting subsequent rounds of Adaboost based directly on the features' weak classifiers (one feature per weak classifier in the original Viola-Jones method), they used previous layers of the cascade as weak classifiers. This resulted in fewer weak classifiers per cascade and therefore a more efficient cascade. They also modeled the detection computational cost of a cascade. The results showed that cascades are computationally more efficient than a single stage classifier, but after adding a certain number of stages there was no improvement. A larger number of stages in the cascade increases the chances of missing hits, and therefore, there is an optimum number of stages for a certain cascade.

2.2.5 Multiresolution Analysis in Feature-based Systems

Methods using either invariant or non-invariant features should have different approaches regarding translation, scaling, rotation, lighting conditions, and articulation. As invariance is usually limited to a few transformations (there is no ideal feature set that is absolute invariant to all transformations), even invariant features often need to be computed over sub-sets of pixels (such as sub-windows at a specific position, scale and rotation). To be able to compute features on the region where the image of the object is, one needs to compute the feature set in many sub-windows using many different scales and positions. An exhaustive search is usually impossible due to the real-time constraints.

The Haar-like features used by Viola and Jones (2001a) can be made invariant to scaling (by dividing the feature value by the area of the feature), but not to any other transformation. This requires the use of a multi-scale approach where each sub-window is a rectangular sub-set of the image. Multiresolution analysis would usually require an image pyramid approach, which is computationally expensive.

Next, it is discussed how to assess sub-windows of a frame in view of translation, scaling, and rotation. For Haar-like features, rotation is limited to specific angles, while translation and scaling are only limited by accuracy and speed.

- **Translation**

In order to detect an object of the same size as the kernel, several sub-windows are examined. If the original kernel has a size $N \times M$ pixels and the image has a size $W \times H$ pixels, then the number of sub-windows S is:

$$S = \frac{(W - M) \cdot (H - N)}{t} \quad (2.6)$$

where:

- W, H are the width and the height of the image
- M, N are the width and the height of the kernel
- t is the translation factor in pixels

A common problem is the fact that classifiers can hit the same object more than once. This happens because two different sub-windows that are very close to each other can yield values that are within the margins allowed by the classifier. Usually, some form of post-processing is necessary to eliminate these additional hits and compose a single coherent hit. Two approaches are possible. The first approach, used in the OpenCV library, is to eliminate little hit regions inside other larger hit regions and take an average for the final hit position. The second approach, is to take into account how close each hit was from the final classifier threshold, and it assumes that the actual hit position is the one with the best threshold.

- **Scaling**

The smallest sub-window is of the size of the images with which the classifier was trained. This base window is called a *kernel*.

Scaling is necessary to find the objects with different sizes from the trained kernel. Once a classifier is trained, there is no need to scale down the image to be assessed. Instead, Haar-like features can be computed directly from the SATs, once for each frame (Viola and Jones, 2004). One drawback occurs due to the rounding process, the discrete nature of digital images causes scaling to generate fractional positions and sizes. Lienhart, Kuranov and Pisarevsky (2003) showed how to compute correction factors that minimised this problem.

Computing every possible scale is not feasible if the real-time constraints are to be met. A reasonable number of sub-windows have to be neglected. Typically, scales

are computed using factors from 1.1 up to 1.4. The smaller the factor, the more demanding the computation. If the factor is too large, objects may be missed. The total size of sub-windows that have to be assessed is :

$$\sum_{i=0}^n (W - M.f^i)(H - N.f^i) \quad (2.7)$$

where:

- W,H are the width and the height of the image
- M,N are the width and the height of the kernel
- f is the scaling factor (the kernel sizes need to be rounded to an integer)
- n is the maximum number of times the scaling is computed, limited by:
 $Round(M.f^n) < W$ and $Round(N.f^n) < H$

For example, for a 640x480 pixels frame, with a kernel of size 24x24 pixels and a factor of 1.1, the total number of sub-windows is 4482974. At 15 frames per second, and if each feature needs 8 lookups, approximately 5.3×10^8 lookups per second are needed just for the calculation of features. In practice, translation with scaling are usually computed in steps of more than one pixel. By using large translation and scaling factors, there can be loss of accuracy in the form of missed objects. On the other extreme, small translation and scaling factors can slow down the classification process and present an overwhelming number of detections.

• **Rotation**

Haar-like features are not invariant to rotation and it is computationally expensive to rotate every sub-window and detect all possible rotations. An alternative for dealing with rotation using Haar-like features is to train several classifiers using rotated examples. The disadvantage of this is the added time and effort to train the set of classifiers, but this is compensated by the flexibility and by the control over separate parts of this process. Jones and Viola (2003) suggested that for faces only a few extra classifiers for different angles would be necessary.

In order to deal with the problem of multi-view faces, Lienhart, Liang and Kuranov (2003) implemented the idea of a detector tree (rather than a single cascade classifier). They validated the results with the XM2FDB video database and concluded that a tree structure for the classifier improves both accuracy and performance. Several parallel cascades of classifiers would also work well for objects that could have very different patterns when assessed from different view points.

Summary of Multiresolution Analysis

Various approaches to analysis of sub-windows in images have been described. The performance of detection and recognition systems are related to the total number of sub-windows surveyed. This number depends on various factors that have to be tuned for best performance or best accuracy. The factors are:

- Kernel size (base sub-window)
- Translation factor
- Scaling factor
- Number of simultaneous classifiers
- Number of SAT

2.2.6 Lighting Conditions

The way an object appears may change dramatically with variations in the environment. Lighting conditions may vary not only due to changes in the light sources, and other objects producing shadows over the detectable object, but also because of the automatic gain control and shutter of a digital camera. Haar-like features can yield good results as long as the positive examples represent different lighting conditions. However, if new situations arise, detection is difficult. Lienhart and Maydt (2002) used an extra SAT to compute the sum of the square of the pixels. With this extra SAT they implemented a simple form of contrast stretching based on the variance:

$$\bar{i}(x, y) = \frac{i(x, y) - \mu}{c\sigma}, c \in \mathfrak{R}^+ \quad (2.8)$$

The Haar-like properties allowed the computation of the stretched feature values without changing the image itself. The mean and variance can be obtained for each sub-window directly from the SAT and an extra square SAT.

2.2.7 Articulated Objects and Motion Features

The current limitations of Haar-like features are mostly concerned with images of articulated objects such as hands. Haar-like features are very robust as long as the object presents a stable shape. Articulated objects cannot benefit directly from this method, unless several parallel classifiers are used for different shapes. Considering that many classifiers are already needed for rotation for a single viewpoint, it would be infeasible to represent all possible shapes where the articulation is too complex. A possible approach is to limit the set of possible articulations. Training of the classifiers may consider each

articulation to be a different object. There is a potential advantage in this approach, since standard gestures could be individually recognised. In a video sequence, where a person gesticulates, the order of two or more recognised articulations could be interpreted as a certain meaning. However, the number of classifiers necessary for a more generic application still poses a problem for real-time application using a single processor.

The original Viola-Jones Method was not very successful in detecting pedestrians. The patterns present in pedestrian detection can vary considerably with lighting and clothes, as well as articulation (non-rigid object). Another factor, was that while in the training of faces, the positive images could be completely separated from the background, that was not the case for pedestrians. Viola et al. (2005) proposed additional filters that took motion into account to try to overcome these difficulties. These modifications were reported to be very successful to track pedestrians in situations where the pure Haar-like feature-based classifier could not cope very well.

Motion filters operate on five images (called motion images) obtained from a pair of contiguous images from the video sequence. The five images are created by computing the absolute difference between each pair of frames. The first motion image (D_f) was computed as a simple subtraction. The other four motion images (U, D, L and R) were created by shifting the second frame by one pixel to the right, to the left, to the top, and down, respectively.

Four types of filters were created by using the five motion images. They are able to generate thousands of possible features in an analogous way as the Haar-like features. The first filter type subtracted the first motion image from the other four motion images, extracting information about the movement. The second filter type was similar to the Haar-like features applied to one of the motion images (U, D, L or R). The third filter type measured the magnitude of motion in the images. Finally, the fourth type of filter (called “appearance filter”) was computed from Haar-like features extracted from the first frame. This would represent the shape of the object without any motion. Although these results were not proved to be as generic as the ones for face detection, the results showed that the method worked well with pedestrian detection.

2.3 Parallel Computing

The main reasons for building a parallel computing system are the limitations of the processing speed, memory access speed and the bus speed of a single processor machine. In this section, the parallelisation techniques in view of Computer Vision systems are discussed.

For some time, it was believed that only very few classes of problems would benefit from parallelisation due to Amdahl’s law implications. However, many practical problems can benefit from parallelisation if the communication between nodes can be minimised

(Wilkinson and Allen, 1999). In image processing and computer vision the work tended to be focused on special hardware (such as FPGAs). Very little work has been done with commodity hardware like Beowulf clusters.

In the next subsection, the parallel approach is discussed, focusing on computer vision using the methods described in the first part of the chapter.

2.3.1 Types of parallel machines

The idea of parallel computers is that one or more tasks are divided into as many processors as possible in order to, either minimise the response time, or to solve a problem that occupies too much memory to fit in a single processor system (usually due to restrictions on memory size or memory access). There are basically two ways of dividing a task, by function decomposition or data decomposition. In data decomposition, the same algorithm is repeated by each individual node, but each has a portion of the whole original problem. In function decomposition, each process makes a portion of the computation using the whole of the data available.

A Shared Memory Multiprocessor System consists of processors sharing a single or various memory modules. The interconnection between the processors and the memories is done via bus or interconnection network. Modern operating systems use virtual memory, and this concept can be extended to a shared memory multiprocessor. The main advantage of this type of implementation is that it is simple to program. However, the hardware is especially difficult and expensive to implement when there are many processors on the same board.

A cheaper hardware alternative is a Message-passing Multicomputer which is created by connecting commodity computers through a network. In this kind of parallel computer each node has its own memory, and therefore it is not directly accessible by any other processor. The network is used to send and receive messages (data or partial results) from other processors. The main advantage of this kind of arrangement is the price and scalability. The system can grow according to the needs. In the past, the main disadvantage was the programming, which involved a lot of effort from programmers to avoid interprocess communication mistakes. In the last few years, several good library implementations (such as MPI) minimised the problem, allowing relatively easy programming. The message-passing paradigm requires that an efficient communication infrastructure is available. Even the best network technology available imposes some performance limits. This is compensated by the fact that there is no need for special hardware mechanisms to control simultaneous access to memory, as occurs with a shared memory system.

2.3.2 Beowulf Clusters

A cluster of computers is not a new idea. For many years, even the major computer industries had some kind of cluster available for specific applications. When the RISC computers were launched, a number of suppliers made available SMP machines connected via a dedicated network interconnection (Aoyama and Nakano, 1999). Although most clusters available at that time used some brand of Unix, they were built based on proprietary hardware and software, and were expensive to buy and to maintain. The idea of simple PCs connected via standard network infrastructure was only recognised as a serious option in the last decade. The original Beowulf cluster project started in 1994 with a few old PCs and the idea quickly spread (Sterling et al., 1995*a*, 1999). The characteristics that made Beowulf clusters an attractive alternative to other commercial parallel systems were price/performance, scalability, availability of open source software, and re-usability of previously developed parallel code. Sterling et al. (1999), Bookman (2003), and Sloan (2004) discussed the details of building and maintaining clusters focusing on Linux operating system to support it.

Nowadays, Beowulf clusters built from commodity hardware are very common. The performance ranges from a few Gigaflops up to a few Teraflops. At Massey a small 16 node cluster was built in 1999 using Linux and Pentium III monoproductors (Grosz and Barczak, 2000). Although the performance was very limited by the Ethernet technologies available at the time, the CS group gained experience in dealing with them. In 2002, a more powerful cluster was built, making use of the now commodity Gigabit Ethernet. The cluster achieved the 304th place in the top500 list (Barczak et al., 2003).

2.3.3 Cluster Topology

The ideal, although usually impractical, way to link the nodes is an exhaustive connection. In this mode, each node is connected to every other node in the cluster. Hence n nodes are connected with $n - 1$ links from each node. There are a total of $\frac{n(n-1)}{2}$ links. In practice, a fully connected cluster is not economical. Other possibilities for the topology of a distributed memory computer were discussed by a number of authors (see for example Wilkinson and Allen (1999)). Historically, there were four important network links: the line, the mesh, the hypercube, and the tree network. Currently, the choices to build economic machines are limited by the restrictions of off-the-shelf components (such as motherboards, NICs, switches etc).

With cheaper switches, which virtually eliminated Ethernet collision problems, the topology models have more theoretical than practical implications for Beowulf Clusters. Three key factors that may limit the overall performance of a Cluster are the network bandwidth, the network latency and the cost of communication infrastructure.

2.3.4 Bandwidth and Latency

Bandwidth is defined by the rate in which data can be transmitted from one node to another. Bandwidth is affected by the hardware (transmission speeds) and by the protocols (reliable protocols cost a percentage of the available bandwidth). For the scope of this thesis, bandwidth is given by the the average data transmission rate (nominal bandwidth less the overheads imposed by the protocols and operating system).

Latency is defined as the time delay to start communication. Again, latency is affected by the hardware and by the protocols. It is difficult to measure latency and bandwidth in isolation. A practical way to measure latency in clusters is to measure the time to transmit an empty message.

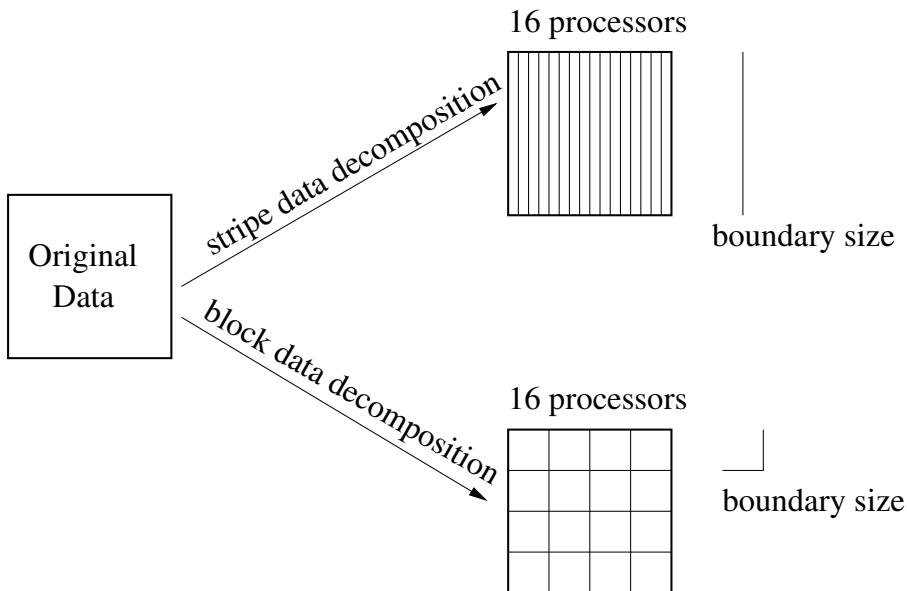


Figure 2.8: Data decomposition: a block data decomposition has a boundary half as large as the stripe data decomposition.

Communication time should be minimised. In many cases, changing the data decomposition strategy changes the communication patterns. For example, in a heat transfer problem the results for the boundaries of the decomposed data have to be sent to the neighbour nodes. The data decomposition could be done contiguously (in stripes) or discontinuously (in blocks). For a 16 node algorithm the block decomposition is faster than that for a stripe decomposition (figure 2.8) because the smaller boundaries imply shorter communication times between the nodes.

The effect of the Latency can also be minimised via a strategy called “latency hiding” (Wilkinson and Allen, 1999; Gropp, Lusk and Skjellum, 1999), where the processors are kept busy for as long as possible. The use of non-blocking MPI calls or persistent communication are different methods to achieve this. The use of threads may also help in some

cases.

2.3.5 Speedup in parallel computers

Most authors in the parallel computing area define a speedup factor (Wilkinson and Allen, 1999), which defines the application's relative performance running in either a single processor or a multiprocessor system. It can be written as

$$speedup = \frac{T_s}{T_m} \quad (2.9)$$

where T_s and T_m are the execution times for single processor and multiprocessor, respectively.

It is expected that, even for the best implementation of a parallel version of a certain algorithm, linear speedups would rarely be achieved. There are many reasons, but the most common is the extra communication effort to synchronise or to gather data from neighbour nodes due to unbalanced loads. Depending on the algorithm, there may be also extra tasks to be done. For example, if 16 processors are working on segmentation of a partitioned image, one object may be split in different nodes, requiring an extra step to join the pieces.

Even if the above mentioned problems do not occur, there is a theoretical limit for the maximum speedup known as Amdahl's law. It imposes a limit to the maximum speedup given that all programs may have serial portions and parallelisable portions. If the program has a considerable percentage of the time on a serial section, it may not be worth to parallelise it, as the speedup may be very low. On the other hand, if the program in question has a fairly small serial portion, then the speedup could be close to a linear one. Amdahl's law may be written as

$$MaxSpeedup = \frac{n}{1 + (n - 1)f_s} \quad (2.10)$$

where n is the number of processors and f_s is the portion of the algorithm that cannot be parallelised (or the serial portion as a fraction of the total). Amdahl's law was widely accepted as a real limitation for parallel computers. An analysis of the Amdahl's formula presented above showed that the serial portion is considered independently from the problem size. Gustafson observed that, in practice, a problem of larger size could be solved without increasing the serial portion runtime (Wilkinson and Allen, 1999), making the percentage of the serial portion a function of the problem size. In that case, an alternative formula (called Gustafson's law, or scaled speedup) is:

$$MaxSpeedup = f + (1 - f)n \quad (2.11)$$

where n is the number of processors and f is the portion of the algorithm that cannot be parallelised. Wilkinson and Allen (1999) illustrate the effects of this law by saying that, for an algorithm that has 5% of serial section and 20 processors, Amdahl's law results in a maximum speedup of 10.26 while Gustafson's law results in a 19.05 speedup.

2.3.6 Message passing paradigm

Message passing libraries were created to facilitate parallel programming for clusters. A message passing library provides communication methods and process initialisation methods and has been the most common paradigm used amongst Beowulf cluster programmers.

The most popular message passing is MPI (Message Passing Interface). The three important advantages of MPI are portability, performance, and availability in different platforms. A full description of the MPI is found in Gropp, Lusk and Skjellum (1999) and Gropp, Lusk and Thakur (1999). Proprietary implementations for specific hardware are also available, see for example, Aoyama and Nakano (1999).

MPI has communication calls for point-to-point communication and for collective communication. Point-to-point messages establishes communication between two nodes, while collective communication does the same for a group of nodes. Messages can use either blocking or non-blocking calls. Blocking messages make the node idle until the communication is completed. Non-blocking messages release the processor to carry on other tasks that are independent of the communication. Non-blocking messages need a system call to test for completion of the communication.

One of the problems with message passing for real time applications is latency. For example, some of the MPI calls use reliable protocols that might put the processor in a wait state until an acknowledgement is received. Unless the algorithm requires little communication, the efficiency of the parallelisation can be very low. When little communication is needed and a simple decomposition is feasible, the resulting parallelisation is called "embarrassingly" parallel. High speedups are achieved in these cases. An alternative to that is the improvement of performance using special protocols. For example, Balaji et al. (2002) used sockets to obtain low latency and high bandwidth, at the expense of the loss of portability.

2.3.7 Parallel Image Processing

Low level image processing is somewhat simple to parallelise. Wilkinson and Allen (1999) discuss many of the low level image processing and their suitability for parallelisation. Simple image processing can usually be solved with an "embarrassingly" parallel approach. If

the algorithm allows each processor to work independently and no communication is necessary until the final result, high efficiencies are obtained with a simple data decomposition. Some problems may require an overlapped boundary.

Higher level image processing usually needs better parallelisation strategies, as partial calculations from neighbour nodes may be needed before carrying out other computations. The amount of data and the frequency of communication between two or more processors depends, intrinsically, on the nature of the algorithm.

Image Processing in Clusters

Barbosa and Padilha (1998) studied how to find the ideal number of processors to obtain the minimum processing time for a certain image processing algorithm. Their tests run on a network of personal computers using edge detection and histogram algorithms. Amdahl's Law was used to explain the times obtained during the tests. They concluded that it is possible to compute the ideal number of processors to execute a certain algorithm as long as the algorithm's parameters (related to the communication and computational patterns) are known in advance. In practice, however, the communication patterns can be more complicated than the two algorithms tested and, therefore, it is very dubious that a simple formula could generalise the behaviours of more complex algorithms. Their model did not consider classification operations required for object detection/recognition.

Segmentation is a very common operation in computer vision. The watershed transform is a popular yet expensive method of segmentation. The pixels are represented as topographic landscape and imaginary "water" is poured into the image forming watersheds. These watersheds define regions and separate objects the same way mountains separate two valleys. Moga et al. (1998) used a simple data decomposition for watershed segmentation and reported good results, although the speedup quickly decreased when using more than 8 processors. Moga (1999) changed the parallel approach by adding a multiresolution analysis step and obtained good speedups, using up to 32 processors.

Another watershed algorithm parallelisation was presented by Roerdink and Meijster (2000). They discussed the two possible decomposition models for watershed transform. In data decomposition, the efficiency depended on the synchronisation of the the neighbouring partial results, as parts of the image of the same object could be sent to more than one processor. In functional decomposition, the efficiency depends on the number of local minima and on the sizes of the corresponding basins, in which case load balancing posed problems. Their conclusion was that watershed transform "remains a global operation, and therefore in the case of parallel implementation at most modest speedups are to be expected". This is an important indication that the key for any successful parallelisation lies in the possibility of minimising the communication among the nodes.

Video Processing in Clusters

A real-time parallel processing system was built by Arita et al. (1999, 2000); Arita and Taniguchi (2001). They used a cluster with Myrinet (with a latency of about 500ns and a bandwidth of 1.28Gbps) and created a whole new protocol (called RPV - Real-time Parallel Vision) to deal with the real-time constraints that standard MPI or PVM cannot deal with. Their system needed to synchronise three cameras in order to obtain 3D images to be analysed. They discussed four main schemes for the parallelisation of computer vision systems, shown in figure 2.9. A brief description and a discussion considering the algorithms described for object detection follows.

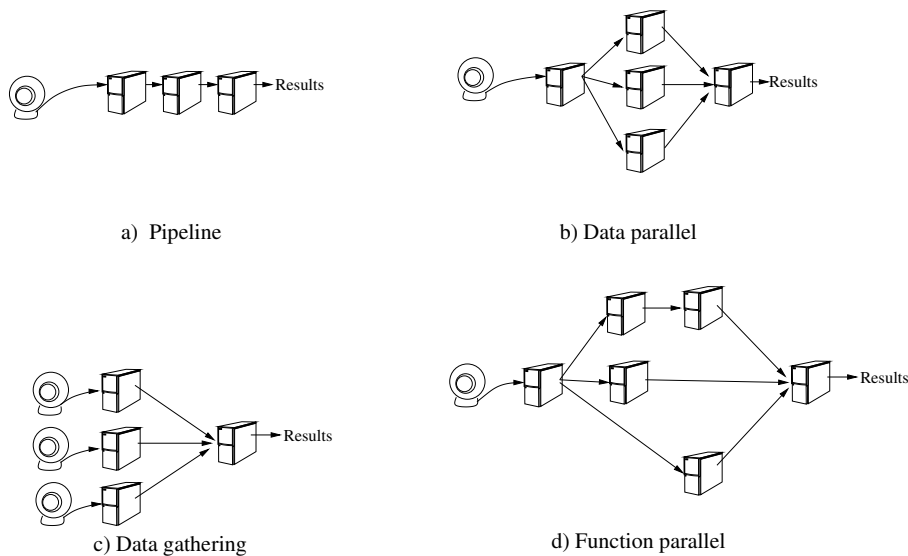


Figure 2.9: Four schemes for Parallel Computer Vision, according to Arita et al. (2000).

- **pipeline parallel processing** - A frame is acquired and processed serially. Each processor deals with the responses of the previous processor.
- **data parallel processing** - Uses simple data decomposition. The frames are split into the processors and the results are gathered by one processor at the end.
- **data gathering** - Each front processor receives a frame, processes it and the results are gathered by one processor at the end.
- **function parallel processing** - Each frame is spread to multiple nodes, so that different algorithms can run independently and their results unified at the end.

In all the four cases, the obstacle for linear speedups is the communication infrastructure, despite improvements that have been made to standard Ethernet adapters and switches (e.g., Gigabit Ethernet).

The pipeline model would be suitable for repetitive processing where some delay would be acceptable. The more nodes are added, the longer the delay. It achieves high throughputs, but latency becomes a problem after a few nodes are added to the pipeline.

Not all computer vision algorithms would work well in the data parallel scheme. A typical example is an algorithm that uses multiresolution analysis. In a multiresolution analysis, if the original frame is divided fairly, some of the sub-windows are split in two or more portions among the processors. In this case, this scheme needs an overlapping boundary, which is usually inefficient. Also, if a classifier is involved in the processing, it is difficult to optimise the system because certain regions of the image would require more processing than others.

The data gathering model could be very similar in performance to the data parallel model. However, the node that gathers the information from the other nodes can become a bottleneck in the system. If the main part of the algorithm needs the pre-processed data from the nodes attached to the camera, then the receiving node needs to be more powerful than the other nodes. Arita et al. (2000) used this scheme because it is suitable for systems with multiple cameras.

In the function parallel approach, it would be very difficult to achieve load balancing. The dynamics of the algorithms do not necessarily fit the available hardware. The function parallel model would suit well systems that need to process independent instances of algorithms, using the data from a single source. Each frame could be broadcast to the nodes. If it is a true broadcast, the communication times would be practically limited by the bandwidth. This model would demand the use of low level protocols, possibly without any of the extra costs of reliability and fairness, in order to achieve good speedups.

In summary, of the four models, the function parallel would be the most suitable for object detection from a single image stream. Load balancing can present a challenge, except if the nodes run instances of the same algorithm with the same data.

2.4 Conclusions of Literature Review

The topic of the Rapid Object Detection was discussed, showing that the applications based on Viola-Jones are, to some extent, successful. However, limitations and gaps in the knowledge regarding the generalisation of the method were identified.

Several training parameters may influence the accuracy of the final classifier. The size of the kernel, the number of stages in the cascades and the variant of the AdaBoost algorithm.

The training images are themselves a source of uncertainty. At least three problems with the training set (positive and negative sets) were identified. These issues are discussed in chapter 3:

- Marking the images using only square kernels.
- The alignment of the positive images, a problem that was also described in early stages in Rowley (1999). Misalignment could cause Haar-like features to be located in different parts of the image yielding a poor quality classifier. Human faces are relatively easy to align, as they have strong geometric characteristics such as two dark spots in the eyes location, but that is not necessarily true for other objects.
- The number of images used did not necessarily translate to good accuracy.

At detection, the following limitations were identified:

- **Occlusion and lighting:** Variation on the lighting condition and partial occlusion may cause the cascades to completely miss objects. Is it possible to yield more generic classifiers based on a modified positive set? The case of partial occlusion for face detection is discussed in chapter 4.
- **Articulation:** How to generalise detection when the object is not rigid? Parallel cascades are a possibility that are explored in this work. The case of hand detection is discussed in chapter 5.
- **Rotation:** Haar-like features are not invariant to rotation. Objects can be missed when they are under rotation that was not predicted at the training phase. Rotation for Haar-like features is discussed in chapter 6. A new feature extraction method based on invariant features is discussed in chapter 7.

The parallelisation of these algorithms has to be carried out in such a way that it minimises communication between the processors. The Viola-Jones approach is suitable for parallelisation because very little communication is needed and the cascades can be trained to work independently. The tasks related to detection using a single cascade classifier, however, requires typical serial processing that makes it difficult to achieve high efficiencies. In many circumstances, more than one classifier is needed (either to detect different objects in the same scene or to detect objects at different angles), and specific strategies to cope with multiple classifiers are discussed in chapter 9.

Chapter 3

Methods and Preliminary Experiments

This chapter presents the methods and the equipment used to collect data for the experiments carried out for this thesis. Section 3.1 discusses implementation aspects of the training and the detection algorithms, as well as methods for collecting sample images. Section 3.2 presents preliminary results of experiments carried out with face detection. Section 3.3 discusses practical issues regarding the implementation of Beowulf clusters. Finally, section 3.4 examines the performance of two clusters built for the Institute.

For the detection experiments two desktop machines were used with the following specifications:

- Dual Core AMD Opteron(tm) Processor 170 2.0 GHz, cache size 1024KB, 2GB RAM, 4023.85 BogoMips¹ (each processor)
- Intel(R) Pentium(R) IV CPU 2.53GHz, cache size 512 KB, 500MB RAM, 5072.48 BogoMips

The cluster node specifications were:

- Dual AMD Athlon(TM) MP 2100+ Processor, 1.7 GHz, cache size 256KB, 1GB RAM, 3457.02 BogoMips (each processor)

The image acquisition process was done using a cheap, but good quality, Logitech 4000 web camera (Philips chipset), which can deliver 30 frames per second for a size of 640x480 pixels.

¹BogoMips is performance benchmark commonly used in Linux

3.1 Implementing the Rapid Object Recognition Training algorithms

This section discusses two training algorithms, the marking process used to gather positive face images, and performance measurements for object detection. For the experiments carried out in Chapters 4 and 5, the original OpenCV implementation of the Viola-Jones methods was used. However, for the other chapters, where major modifications on the original method were made, a new implementation was coded in order to accommodate the changes. The AdaBoost algorithm has problems with convergence for certain training sets, and a simple modification to cope with the problem is proposed in this section.

A simple Convex Hull based classifier is described. This type of classifier has a limited accuracy, but it can be used to pre-process a large portion of sub-windows.

The methods to collect sample images depend on manual marking. The methods used to mark faces and hands are briefly described. Finally, the methods and databases used to measure accuracy of classifiers for object detection are presented.

3.1.1 Adaboost: a simple modification for convergence and speed

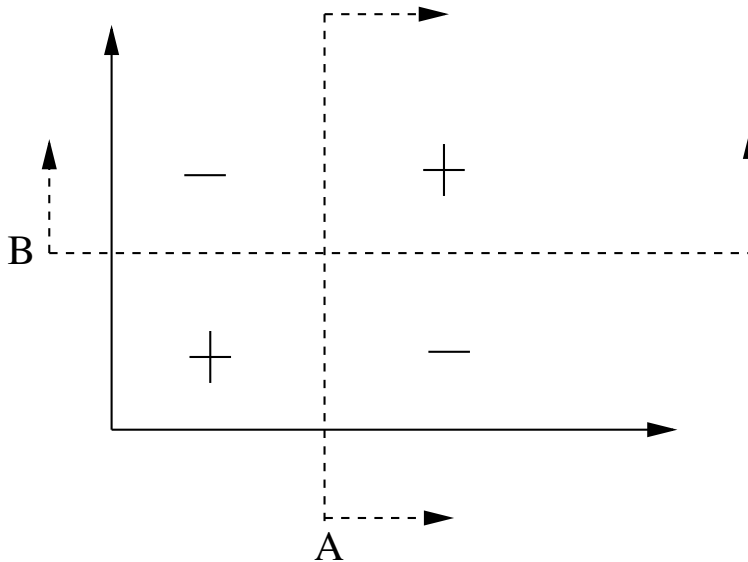


Figure 3.1: Classification using thresholds does not converge.

Recently, a few problems with the convergence of the AdaBoost algorithm have been reported (Rudin et al., 2004). It has been known, for some time, that training with certain sets do not converge when using simple threshold as weak classifiers. As an example, let's consider the figure 3.1. This figure shows two very small positive and negative sets. By trying to set simple thresholds there are four possibilities to classify correctly each element. It is shown that there is no solution in which the training error converges to

zero. Assuming that two thresholds, A and B, are used to separate the positive and negative sets, the hypothesis would be as follows:

$$H(x) = \begin{cases} 1 & a.h(x) + b.h'(x) \geq 0 \\ -1 & otherwise \end{cases}$$

For a correct classification, one would need to solve the following system, for which there is no solution for a and $b \in \mathfrak{R}$:

$$-a - b < 0$$

$$-a + b \geq 0$$

$$a - b \geq 0$$

$$a + b < 0$$

The effect of this characteristic is serious during training. The threshold values go through an infinite cycle, yielding classifiers for which the training error does not improve, no matter how many weak classifiers are added to them. Rudin et al. (2004) analysed the dynamics of AdaBoost and its cyclic behaviour, though many open questions remain. It is not known in which conditions AdaBoost converges or behaves chaotically. More interestingly, it is also not known why it does achieve a strong generalisation performance in most cases. Our own experiments suggests that training large sets with a large number of dimensions tend to produce good classifiers, while training with small sets, with a small number of dimensions, may cause the process to behave erratically.

A possible solution for this impasse is to use more than one stage. Like Viola-Jones method, the hit rate is maximised by changing the stage threshold. Each stage may be limited by either the number of weak classifiers or by a maximum false detection rate. For example, the set seen in figure 3.2 has no solution if a single stage is used. However, by using sub-sets of the negative set to train various stages, it yields a good classifier.

This training strategy is simple, but effective. Depending on the difficulty of the problem, it may take many stages, especially if the positive set data does not cluster in the feature space. If positive and negative samples are located in the exact same point

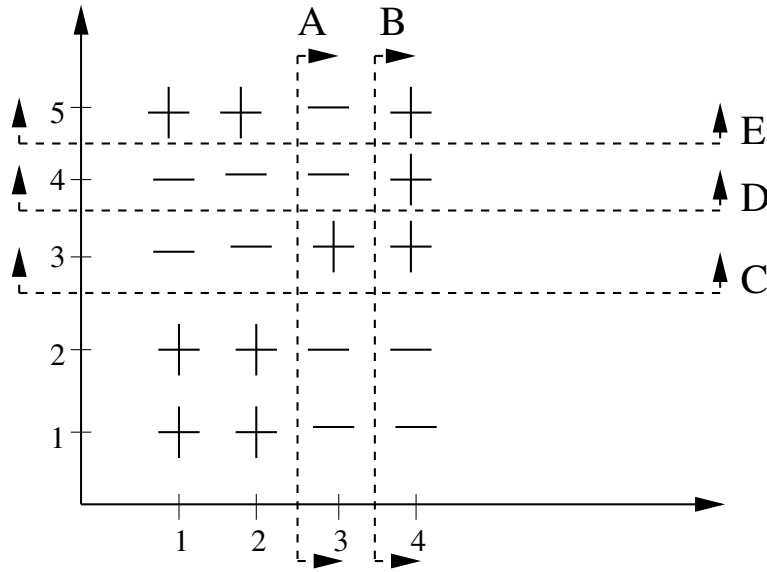


Figure 3.2: This set converges if the the classifier has four stages.

of the feature space, then the training would never converge to zero. Viola and Jones (2001a) used several stages in their method in order to achieve performance, not due to convergence problems. In fact, due to the high number of dimensions of Viola-Jones feature space, AdaBoost converges without problems when using the original Haar-like feature set.

A modified AdaBoost, shown in algorithm 3, was used in this thesis for many of the experiments.² Algorithm 3 is very similar to the AdaBoost version used by Viola and Jones (2001a), but the goal is different. While in Viola-Jones method the cascades were used for performance, here the cascade approach was used to achieve convergence with high positive hits.

When both the positive and negative sets are finite, it is sometimes convenient to force the hit rate to 100%. In order to do that, the thresholds of the stage being trained are changed. This causes the false detection rates to increase.

During the training process, if a certain stage of the cascade does not achieve the hit and false detection specified, negative elements are discarded. The training of that particular stage starts again with a smaller negative set. The discarded negative samples are added later, to the negative set of the next stage. This process of using negative subsets splits the negative set in more feasible portions, facilitating the training. If enough stages are trained, the training error converges to zero.

There is one drawback for this approach. Depending on the distribution of the feature values, it may happen that very few negative elements are eliminated at a certain

²All the classifiers used in the experiment 7, in chapter 7, and the classifiers used in chapter 8, were produced using algorithm 3.

stage. The number of weak classifiers may surpass the number of elements in the training sets, which may yield long and inefficient cascades. This problem can be minimised by clustering the negative set.

Algorithm 3 AdaBoost2(*(value, class), T*)

The input is a set of training examples $(x_1, y_1), (x_2, y_2), \dots, (x_n, y_n)$ where x_i is a feature $\in X$ and y_i is a class $\in \{-1, +1\}$, and a number of rounds T . It outputs a linear combination of weak classifiers $h_t(x_i)$, each a factor α_t .

Initialise weights for each element $D_1(i) = 1/n$ where n is the number of elements

While $P < P_{requ}$ and $F > F_{requ}$:

1. Train a weak classifier h_t (e.g. a simple threshold) using the weights in D_t so that $h_t \in \{-1, +1\}$

2. Compute the error associated with the weak classifier:

$$error_t = \sum_i (D_t(i) h_t(x_i) y_i) \quad (\text{the sum of the elements' weight that are incorrectly classified})$$

3. Compute $\alpha_t = 0.5 \ln\left(\frac{1-error_t}{error_t}\right)$

4. Update the weights $D_{t+1}(i)$ such as D_{t+1} is a new distribution:

$$D_{t+1}(i) = D_t(i) \exp(-\alpha_t y_i h_t(x_i))$$

(the weights decrease if the element is classified correctly or increase if it is classified incorrectly)

$$\text{normalise the weights } D_{t+1}(i) = \frac{D_{t+1}(i)}{\sum_i D_{t+1}(i)}$$

5. After T rounds the final classifier is: $H(x) = \text{sign}(\sum_{t=1}^T \alpha_t h_t(x))$

6. Compute the positive hit rate P and the false detection rate F for round $t + 1$

7. If the $F > F_{Max_per_round}$

restart round t with a sub-set of the negative set (choose N elements (x_n, y_n) where $y_i = -1$)

8. Else

start round $t + 1$ after eliminating all negative samples that have not passed round 0 to t

3.1.2 Building Classifiers with a set of Convex Hulls

Given a set of points P , a Convex Hull is defined as the smallest convex set of points that contains P and is a sub-set of P . The Convex Hull was used before in Pattern Recognition (see for example Akl and Toussaint (1978)). The following algorithm (algorithm 4) is a very efficient one to compute convex hulls (Andrew, 1979):

Algorithm 4 *ChainHull*((*value, class*), *T*)

Input: a set $S = P = (P.x, P.y)$ of N points

Sort S by increasing x - and then y -coordinate.

Let $P[]$ be the sorted array of N points.

Get the points with 1st x min or max and 2nd y min or max

- minmin = index of P with min x first and min y second
- minmax = index of P with min x first and max y second
- maxmin = index of P with max x first and min y second
- maxmax = index of P with max x first and max y second

Compute the lower hull stack as follows:

1. Let L_{min} be the lower line joining $P[\text{minmin}]$ with $P[\text{maxmin}]$.
2. Push $P[\text{minmin}]$ onto the stack.
3. for $i = \text{minmax}+1$ to $\text{maxmin}-1$ (between the min and max)

if ($P[i]$ is above or on L_{min})

Ignore it and continue.

while (there are at least 2 points on the stack)

Let $PT1$ = the top point on the stack.

Let $PT2$ = the second point on the stack.

if ($P[i]$ is strictly left of the line from $PT2$ to $PT1$) break out of this while loop.

Pop the top point $PT1$ off the stack.

Push $P[i]$ onto the stack.

4. Push $P[\text{maxmin}]$ onto the stack.

Similarly, compute the upper hull stack.

Let W = the join of the lower and upper hulls.

Output: W = the convex hull of S .

Algorithm 4 is only suitable for two dimensions. To be able to cope with a higher dimensionality, an approach to combine two dimensions was used, finding a convex hull for each pair of features. At the detection phase, a sub-window's pair of features is examined to see if they are within the borders of their particular convex hull. If any point

is outside the boundary, the classifier considers that as a negative answer, discarding that sub-window. The Convex Hull classifiers only work well if the features cluster into a single cloud in the feature space, otherwise the number of false detections would be too high. Convex hull classifiers can also be used as a pre-processing stage for detection, as it is able to eliminate a number of negative points very rapidly. The advantage of such classifiers is that the time to train and detect is extremely fast.

A sample of a convex hull used in early experiments with OCR is shown in figure 3.3. In this figure, the moment invariants for hand digits zero samples were computed, with the first two moments plotted. The time to train 55 convex hulls with 6000 positive samples took less than a second.

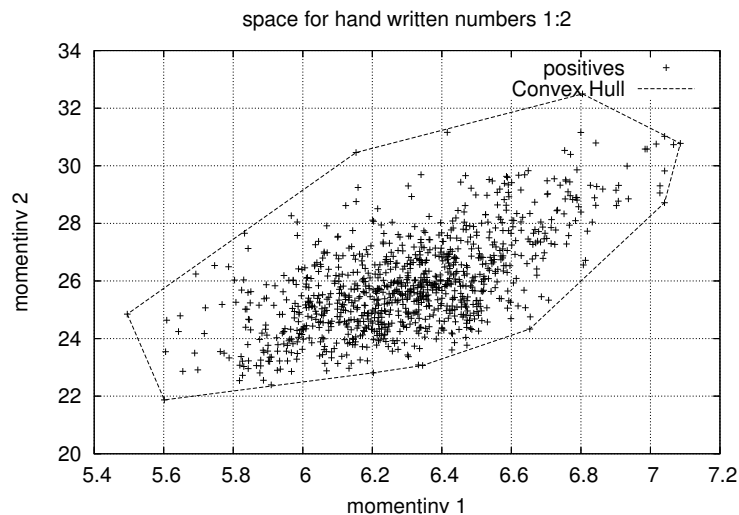


Figure 3.3: Example of convex hull used as a classifier.

3.1.3 Marking the positive examples

In practice, the marking of the positive examples, either to use them in the training or in the testing set, is a tedious job. The same can be said about the negative set, which has to be checked not to contain any image that could be considered positive. In that sense, the job is very subjective and the person who is choosing and marking can negatively influence the training process. Currently, it is not possible to automate the process completely unless a classifier partially trained is already available. For marking faces, it was decided to use a three points method that takes into consideration the fact that people have slightly different shapes and proportions. The marker has to pick three points, two points in the centre of the eyes and one point in the centre of the mouth. With the three points marked, a circle can be found. Its centre and radius are parameters for the rectangle (or square) final marking, which is used to separate the positive example

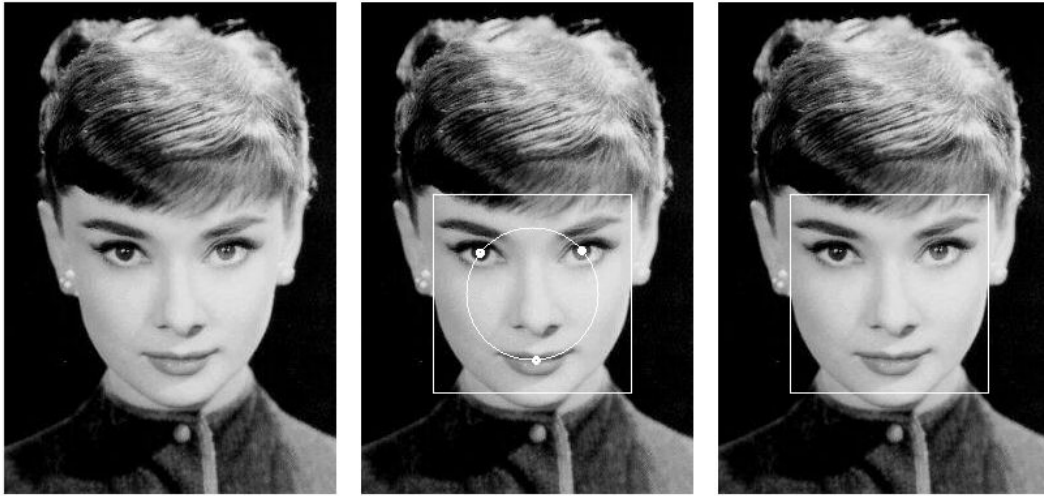


Figure 3.4: Marking a face from the CMU-MIT database

and to create truth tables for testing. Figure 3.4 shows how the marking process works. Typically, the width of the square marker is 1.5 times the diameter of the circle.

3.1.4 Accuracy and Performance Measurements

There are various sources of inaccuracies for detection algorithms, including possible negative images that are so similar to the positive pattern that it affects the learning process. Images rarely present linearly separable features. One could expect a much higher false positive rate and a lower hit rate, when effectively using the classifier with test data. One way of measuring the quality of the classifier is to use a ROC curve (Receiver Operating Characteristic curve, Egan (1975)). Cost curves were appointed as an alternative for classifiers comparison by Drummond and Holte (2006). Different authors may use different methods to plot ROC curves, but it is widely accepted that it is difficult to compare methods by only using a single hit rate (Viola and Jones, 2001*a*). A general discussion on practises for performance and accuracy measurements for computer vision applications were presented by Thacker et al. (2005). In order to measure accuracy in face detection, the most widely used database is the CMU-MIT, which was created by Rowley (1999) as an extension of a previously created database

There is no agreed benchmark for performance (measured in terms of speed or frames per second). However, it is expected that real-time detection delivers, at least, more than one frame per second. Performance varies with certain parameters, translation and scaling factors being the two most important ones. Also it is important to consider some heuristics that might change the performance characteristics. For example, in the original implementation of OpenCV, the function to compute Haar-like features uses half-

resolution images³, which decreases the number of sub-windows classified by the detector. The parameters that affect **training** are:

1. the kernel size.
2. number of samples (positive and negative images).
3. number of stages in each cascade classifier (influenced by the hit rate target and the false detection rate target).
4. Haar-like types (normal only or both normal and tilted).

And the following parameters affect the **detection** phase:

1. image resolution.
2. translation factor.
3. scaling factor.
4. number of cascades.

3.2 Preliminary Experiments

For reference, ROC curves were plotted using the original implementation of Viola-Jones in OpenCV (figures 3.5, 3.6, 3.7, 3.8, 3.9). The tests were carried out using the original version of Adaboost found in the OpenCV library Bradski (2000).

In these experiments, the classifiers were trained with frontal faces only. All the positive examples were extracted from the FERET dataset and the negative examples were extracted from a mix of images that did not contain any faces. The CMU-MIT dataset was used as a test set to plot the ROC curves. This dataset contains many non-frontal faces, as well as cartoon-like faces that are not recognised by our classifiers. For this reason, it is not expected that the classifiers achieve the same level of accuracy of the sample classifiers from OpenCV. The ROC curves in this section were produced with fixed detection parameters:

- scaling 1.2
- translation 1

When using only the upright features (from figure 2.3), the set of features is called BASIC. If the classifiers use both upright and tilted features, they are called ALL (figure 2.3 shows the upright features and figure 2.5 shows the tilted features). The sample

³*haarobjectdetection()* in OpenCV

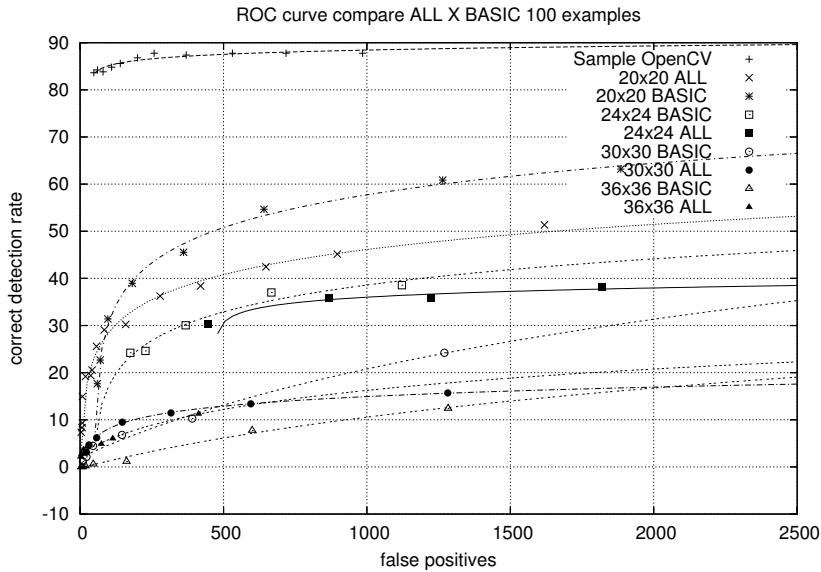


Figure 3.5: ROC curve BASIC versus ALL features, 100 positive examples

classifier from OpenCV used both normal and tilted features, and it was originally trained with a very large face database, including non-frontal faces (Lienhart and Maydt, 2002).

Figures 3.5 to 3.9 show the comparison between using all types of feature (ALL) and only upright ones (BASIC), with kernel sizes varying between 20x20 to 36x36 pixels. Each figure has a different number of training examples. The results showed that when using large kernels the results were not as accurate as the classifiers using small kernels. The larger the kernel the more features the training process has to cope with. Training becomes more difficult, not only because there are more dimensions to compute, but also due to the increasing number of possible images that exist at a certain resolution. At detection, cascades with larger kernels miss some of the faces, because the scale of the face is smaller than that of the kernel.

In terms of feature type, there was no significant difference in accuracy between using only upright features (BASIC) or including tilted features (ALL). However, the classifiers with BASIC needed more weak classifiers than their counterparts using ALL. As expected, classifiers trained with more positive examples presented better accuracies.

These preliminary results guided the experiments carried out in the other chapters. Next, the performance characteristics of Beowulf clusters is presented.

3.3 Building Beowulf Clusters

This section presents some performance results obtained from two Beowulf clusters called “Sisters” and “Helix”. These clusters were built at Massey University in 1999 and 2002, respectively. Issues concerning network latency and the effect of the switching fabric and

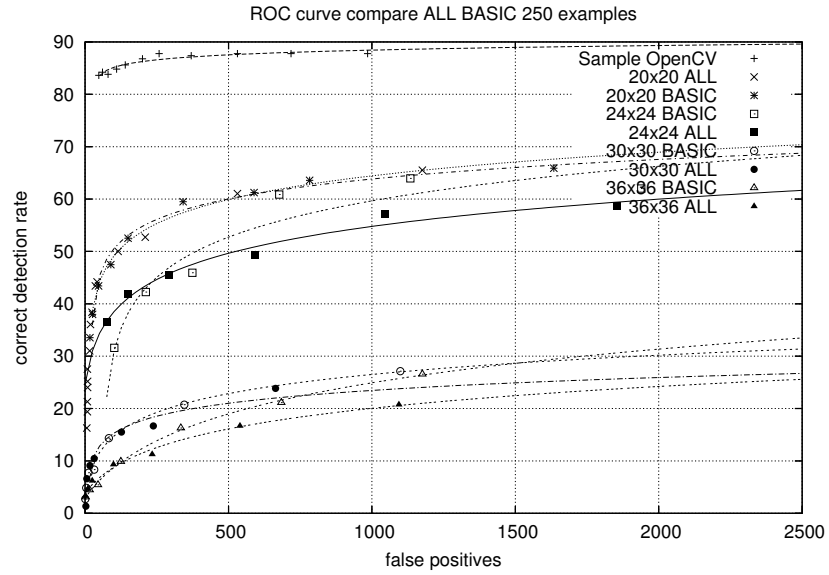


Figure 3.6: ROC curve BASIC versus ALL features, 250 positive examples

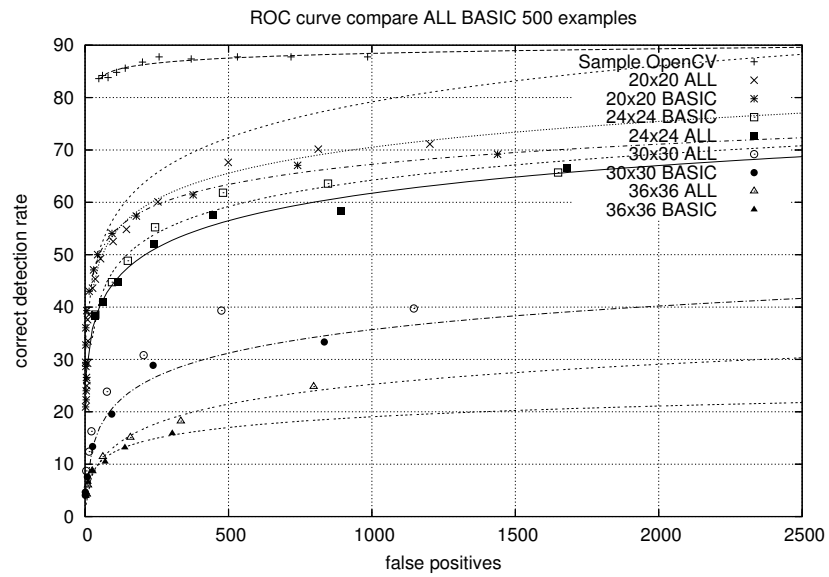


Figure 3.7: ROC curve BASIC versus ALL features, 500 positive examples

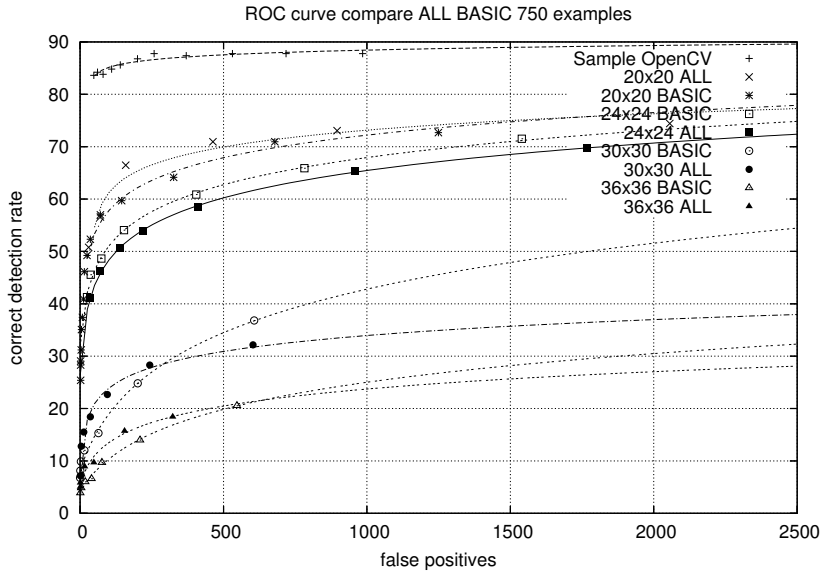


Figure 3.8: ROC curve BASIC versus ALL features, 750 positive examples

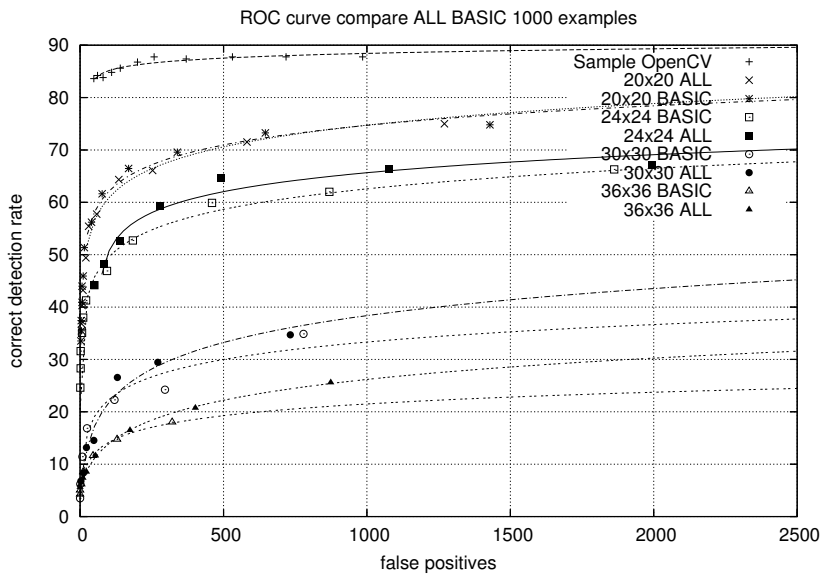


Figure 3.9: ROC curve BASIC versus ALL features, 1000 positive examples

network topology on performance are discussed. In order to assess how the system performed using the message passing interface (MPI), two test suites (*mpptest* and *jumpshot*) were used to provide a comprehensive network performance analysis. The performance of both clusters are compared in this section. These clusters were used to train and assess the performance of all the classifiers produced for this Thesis.

Many benchmarks are available, but few are able to characterise a cluster in a useful and reproducible way. Gropp and Lusk (1999) discussed many of the traps found in performance measurement. The Linpack benchmark was chosen as the floating point benchmark because it would test the scalability of the Helix rather than just the raw peak performance. The macro performance of the machine is illustrated by the Linpack benchmark, but the micro performance of the Helix system also needs to be analysed. In order to do that, the *mpptest* test suite, developed by Gropp and Lusk (1999), was used.

3.3.1 A brief description of the clusters

The “Sisters”

The Sisters was the first Beowulf cluster built at Massey University. It consists of 16 nodes connected by a fast Ethernet network. The nodes used Pentium III CPUs, running at 667MHz with 256MB Ram (they were since upgraded using dual processor nodes). The server is a dual Pentium III with 1GB Ram. There is a single switch linking the main server and the nodes. Key bottlenecks in the system includes the high latency and low bandwidth of the network.

The “Helix”

Considering all available options at the time, the nodes were built with dual Athlon MP2100+ CPUs, 1GB memory and dual 3com 64bit gigabit network interface cards. The dual processor configuration is favoured so that the network interconnect costs would be limited. The communication between the nodes is achieved with standard unmanaged gigabit switches. Each node has two Gigabit Ethernet cards. Theoretically, a dual network increases the available bandwidth. In practice, there are bus contention issues that may prevent the cards from working simultaneously at full speed. The topology is a simple grid, with the nodes arranged in rows and columns, shown in figure 3.10. Each node chooses the best path to the next node based on static tables. Given the constraints, i.e., the number of nodes and the availability of ports on the switches, the connections between nodes need a maximum of 2 hops.

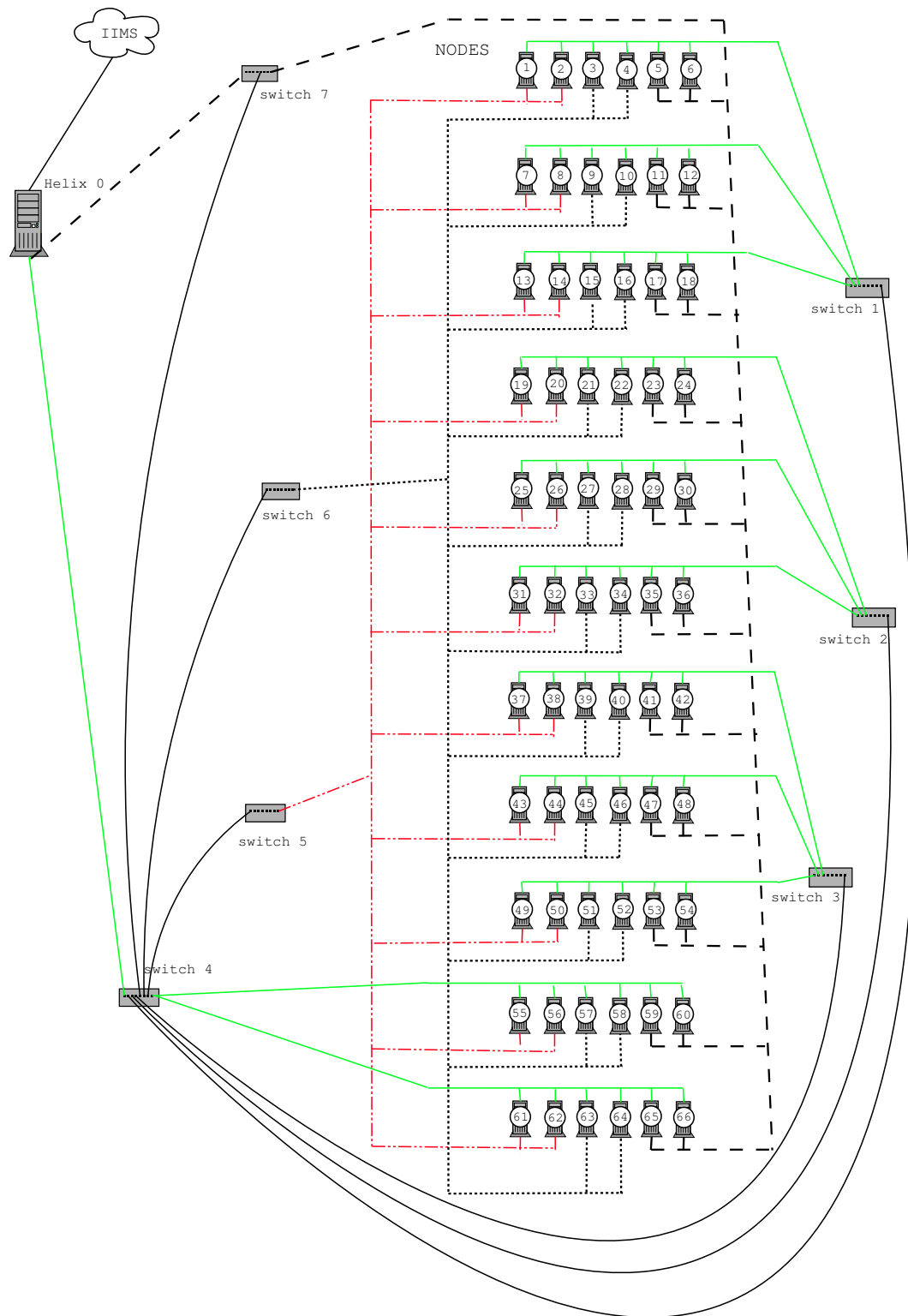


Figure 3.10: The Helix Beowulf-type cluster's topology (built in 2002 for Massey University).

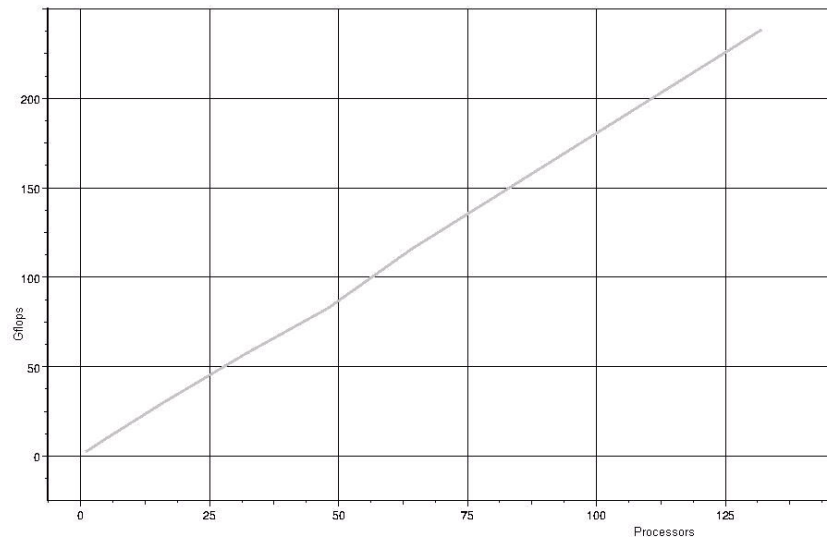


Figure 3.11: Linpack Rmax rating versus number of processors.

3.4 Performance of Clusters

The testing of the Helix performance using the Linpack benchmark and specific issues regarding the network infrastructure (using mmptest and jumpshot) are investigated next.

3.4.1 Linpack performance results

The Linpack benchmark measures the performance of a distributed memory computer while solving a dense linear system in double precision (Dongarra, 1989). Several factors determine the performance achieved on this benchmark, but the bandwidth and latency are a significant component. High bandwidth and low latency ensure that the communication is not a bottleneck. Each of the nodes in the Helix contains 1GB of RAM, so this provides a limit to the maximum problem size. Since 8 bytes per double precision floating point number are required, the maximum theoretical matrix size is 94,118 by 94,118. However, due to operating system overheads, a more realistic size using 80% of the memory gives a figure of 80,000 by 80,000. The theoretical peak performance (Rpeak) of the Helix (the maximum Gflop rating that the machine is guaranteed never to exceed) is 448.8 Gflops, however, this is not a useful measure since it is not achieved even if every machine is working independently. The peak Linpack performance (Rmax) using a problem size of 82080 was 234.8 Gflops (*Top 500 computers*, n.d.).

The performance of the Helix on the Linpack benchmark versus number of processors shows that the system scales almost linearly (see figure 3.11). This is due to the high bandwidth, reasonable latency switching and the grid topology. The switches each have 24 ports (as seen in figure 3.10) and the vertical connections connect up to 22 nodes on a

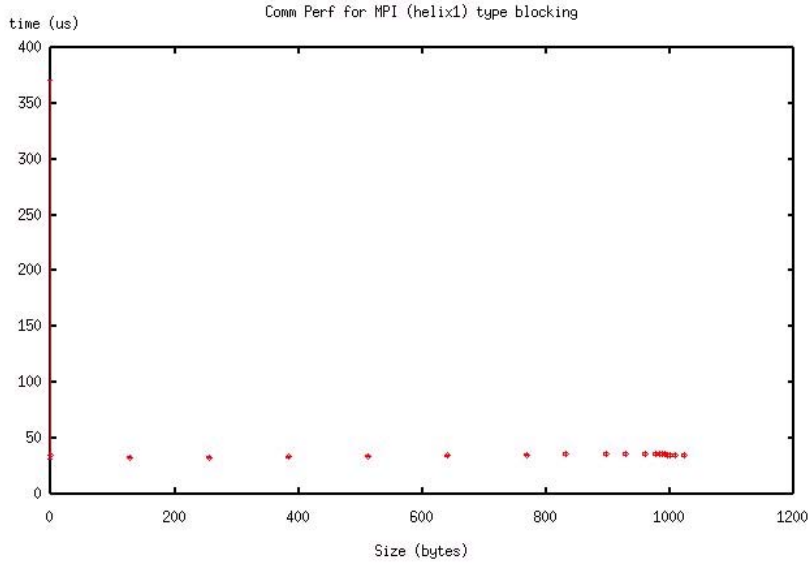


Figure 3.12: Point to point messages in Helix (short, blocking)

single switch. This means that up to 44 processors can be connected with a single high bandwidth hop between processors. When a larger number of processors are required for the Linpack benchmark, then the nodes must be selected to make optimal use of the grid layout.

3.4.2 An analysis of performance using mpptests

This set of applications evaluates how efficiently the hardware is being used by the whole system, including the operating system and MPI (Gropp and Lusk, 1999). There are other benchmarks available that test a variety of MPI calls, for example SkaMPI Reussner (2001).

A point to point test shows that for short messages, in a Gigabit Ethernet, the times are almost constant. For message sizes of up to 1000 bytes, the communication times are around 35 microseconds (figure 3.12). This may be considered as an indication of the latency itself. This compares favourably with the same test run in an older cluster with fast Ethernet. In the cluster called Sisters, as the message size increases the communication times build up quickly, and it goes to 250 microseconds for a message size of 1000 bytes (figure 3.13).

Theoretically, the gigabit Ethernet should be 10 times faster but due to latency effects, the speedup is not as high for small messages. The latency of less than 40ms is reasonable for the gigabit Ethernet system. The communication times for very short messages are about 75% when compared to the fast Ethernet and there is a gain in speed for any message size. The bigger the messages, the more advantageous becomes the use of Gigabit

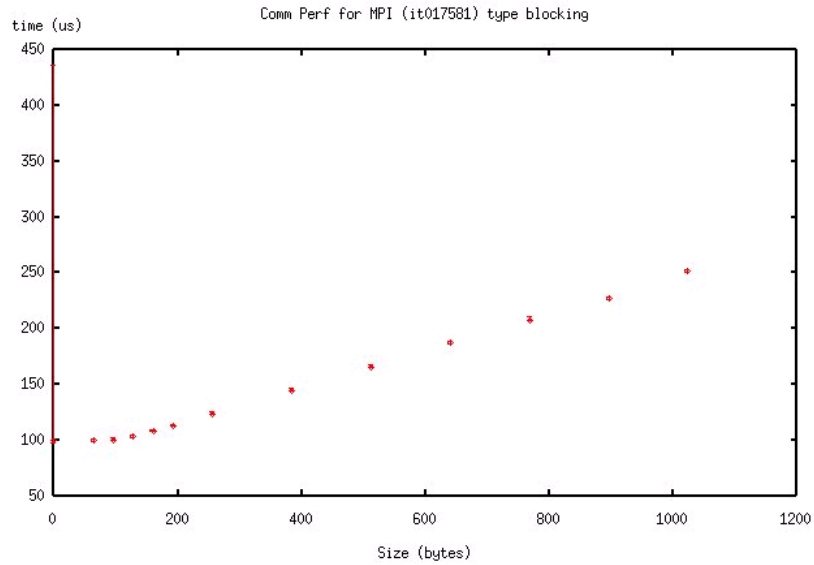


Figure 3.13: Point to point messages in Sisters (short, blocking)

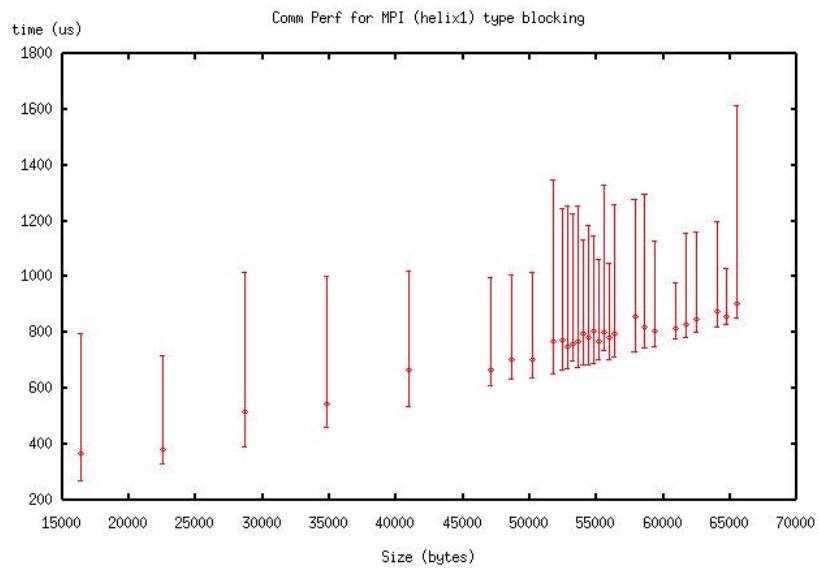


Figure 3.14: Point to point messages in Helix (long, blocking)

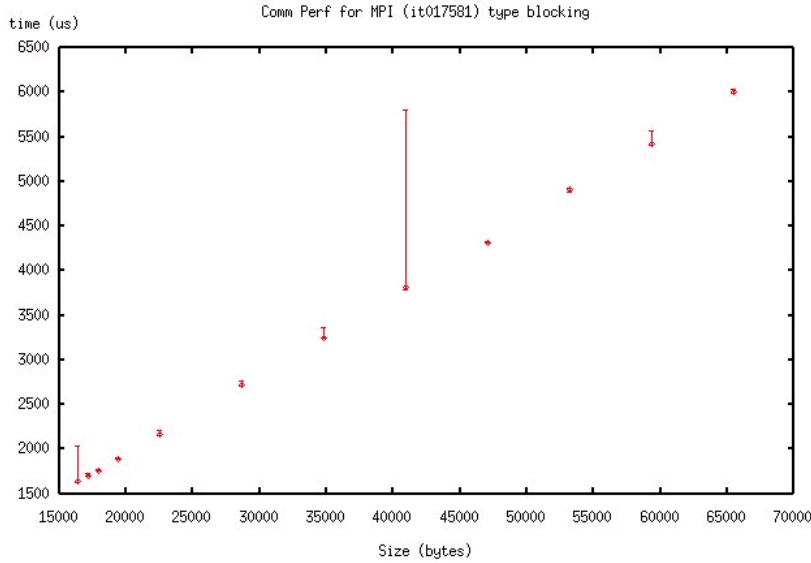


Figure 3.15: Point to point messages in Sisters (long, blocking)

Ethernet. Figure 3.14 shows that the times grows linearly up to 65KB in the Helix. The same tests running on the Sisters (using fast Ethernet) are shown on figure 3.15. For 65KB the transfer times are approximately 6.7 times the ones obtained on Helix. The same trends were observed when using non-blocking/point to point messages. The maximum theoretical bandwidth in the fast Ethernet and on a Gigabit Ethernet are 12.5MB/s and 125MB/s, respectively. For long messages (figures 3.14 and 3.15), Fast Ethernet is approximately 87% as fast as the maximum. A similar analysis for the Gigabit Ethernet shows that it is only about 59% as fast as the theoretical maximum.

Collective operations may present a different challenge for performance measurements due to the particular way they are implemented. Both clusters were measured using *goptest* with *-bcast* option operations. Figure 3.16 shows an example of collective calls in MPI. In this example, one can see that the communication times increase dramatically with the increase of the number of processors.

3.4.3 Ring test using jumpshot

An interesting observation was made by Zaki et al. (1999). They found that the Beowulf network structure can sometimes be revealed by running a ring test application in MPI. Applying this same concept, ring tests were run in the Helix cluster. When two neighbouring nodes are not located in the same switch, it would be expected that a slightly longer message passing time would be measured. For example, there are two hops between nodes 36 and 37 (figure 3.10). Running a ring test with a 127KB message size, a CLOG file was obtained and it is shown in figure 3.17.

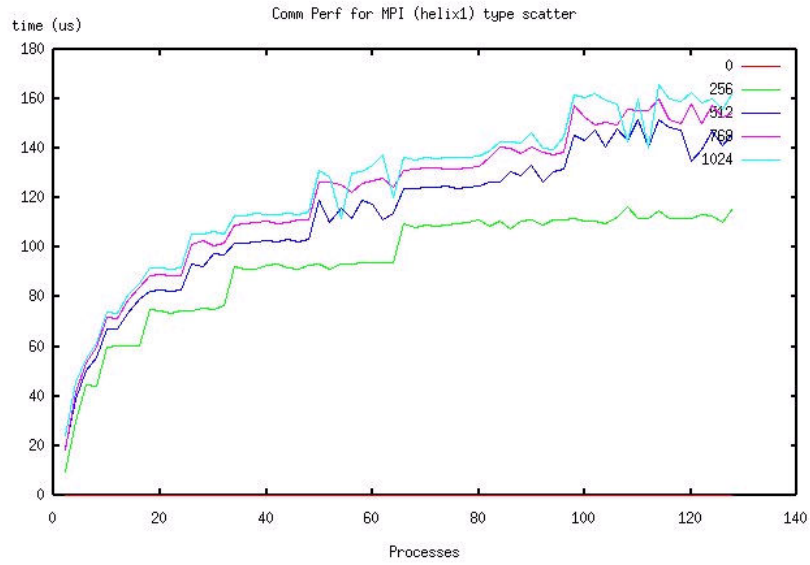


Figure 3.16: Goptest (Broadcast operation) running in 128 processors.

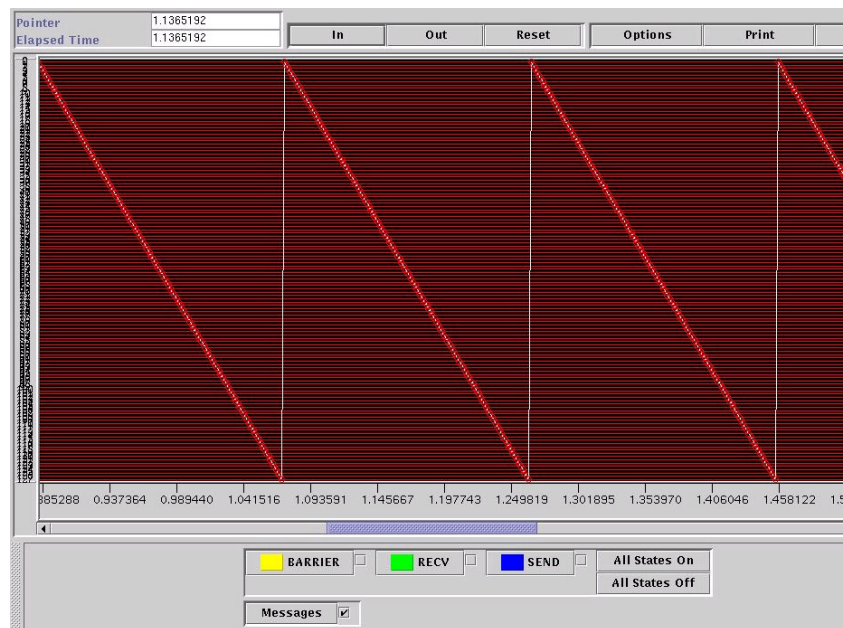


Figure 3.17: A ring test using 127KB message and 128 processors

Figure 3.17 shows that there is no noticeable effect of the topology, even when using relatively long messages. Several tests using the processors in different order were carried out to confirm the results. This figure also shows that the point to point messages are being delivered with little degradation, when making multiple hops over switches. This could be explained by the fact that the additional delay introduced by the switches is small compared to the communication overheads of MPI.

3.5 Summary

This chapter showed the influence of important parameters in the original Viola-Jones method. Generally, the more training samples the better the chances of detection. The influence of the kernel size was that the larger the kernel, the more difficult is to train the classifiers. This was due to a larger number of Haar-like features the training algorithm had to cope with. The influence of the feature type was surprisingly small. A few normal features were able to achieve the same discriminative job of the set that included tilted features.

The parallelisation and Beowulf cluster implementation was discussed in this chapter and performance results showed the differences of the two Beowulf clusters built for the institute. The classifiers produced for this thesis were trained using both clusters. Based on these preliminary results, as well as on the Viola-Jones method communication patterns, a model for the parallelisation of the detection algorithms was proposed and tested (details in chapter 9). A platform for a mobile computer vision system based on Viola-Jones method is also discussed in that chapter.

Chapter 4

Partial Occlusions in Face Detection

4.1 Introduction

In this chapter, the Viola-Jones detector was evaluated in the presence of partial occlusions for face detection. The hypothesis that it is possible to train a single classifier to find partially occluded objects is tested. The experimental results shows that the method can be very robust to partial occlusions and that the hit rates can be improved by generating random occlusions on the positive example set. However, the results also shows that the negative example set must not be neglected otherwise the number of false positives may increase.

The problem of tracking and recognising objects in real-time presents specific challenges, due to the balance between real-time constraints and the accuracy needed. There are additional problems in the presence of partial occlusions and shadows. Even the best algorithms can fail or behave erratically under those circumstances. Figure 4.1 shows typical frames in which faces are partially occluded.

The initial hypothesis was that with the correct positive examples the features would be chosen in such a way that partial occlusions and shading would be treated as part of the object, improving the hit rate when these events occurred. But, what composes a good positive example set of images? It would be infeasible to provide a training set with all the possible variations of partial occlusions. The first idea was to use occlusions from the background. However, these occlusions might get specific characteristics of the background and might not be generic enough. The second idea was to generate occlusions randomly. Both ideas were tried in the experiments described in the next section.

The following section describes how the training process was set up. Next, the details about how the experiments were carried out in order to test the classifiers are presented. Then, in the results section the hit rates for various classifiers and problems related with



Figure 4.1: Examples of partial occluded faces found in the three sets.

false detections are discussed.

4.2 The Training Process

The training process started by choosing a number of images for the positive images set and the negative images set. The division, simple in principle, is in itself subject to errors on both sets due to alignment problems and unexpected similarities between the object and some negative images.

Positive images were gathered from the FERET database (Phillips et al., 2000). Negative images were gathered from the web, from images acquired with the web camera and from image libraries such as the CorelDraw image dataset. The negative images set needed to be checked for the existence of any object that could have an impact on the training process. False positive objects on the negative images may cause the training process to discard good features. This looks like a simple problem, but errors may appear due to resolution changes in the original set. Figure 4.2 shows an example where part of the background in a lower resolution was wrongly classified as a face by an earlier version of the face classifier. The only way to check for false positives is by searching the negative images set manually.

The next step of the training process was the calculation of the Haar-like features over the positive and negative image examples. The total number of features per frame may surpass the total number of pixels, making the computation very intense. Due to these constraints all the images for training were changed to a lower resolution. In these experiments, all the images had a resolution of 24x24 pixels. Then the classifiers were trained using the stump version implementation of AdaBoost available in OpenCV (Bradski, 2002).

Three training experiments were carried out. In the first one, a web camera was used to acquire images of a single person with and without occlusions and shadows (as an example, see the first image to the left of figure 4.3). In the second experiment, FERET

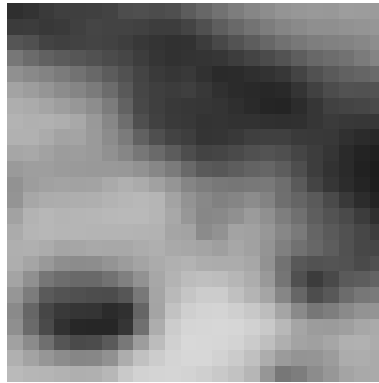


Figure 4.2: Example of a false positive object within the negative set.

Table 4.1: The training parameters for the classifiers.

	Classifier 1	Classifier 2	Classifier 3
Number of positive images	976	4767	13566
Training images	web-cam, 1 person	FERET	FERET
Presence of occluded images?	yes	no	yes
Number of negative images	1134	1134	1134

images were used, all quasi-frontal faces without any occlusions. In the third experiment, modified FERET frontal face images were used to train a classifier with occlusions. Gentle Adaboost with the complete Haar-like features (upright and tilted features) were used in all experiments. The classifiers produced by these training experiments had 30 layers each. Table 4.1 shows the different parameters used in the training processes.

In the first experiment, 976 images of a person were acquired by a web camera at an initial resolution of 352x288 pixels. Random portions of the face were manually occluded with background pixels obtained randomly from images that did not contain any face. The marking process was manual, according to the process described in section 3.1.3. The resulting classifier of this training process was called *Classifier 1*.

The second experiment used 4767 FERET faces for the positive images set. The main reason to have this classifier was for reproducibility purposes and to compare the false detection rates with the other two classifiers. It was expected that this classifier would not be as generic as the original OpenCV sample classifier, because it was trained with frontal faces only. The resulting classifier of the second training process was called *Classifier 2*.

In the third experiment, 1938 of the FERET frontal images were partially occluded with random pixels instead of background pixels. When providing these images with occlusions, one would expect that the Adaboost algorithm would yield a more generic classifier. Partial occlusion and shading effects would be regarded as part of the object. Whenever at least half of the face would be visible, a hit would be expected. The position of each face was marked as before. Based on the position and sizes found in this step,

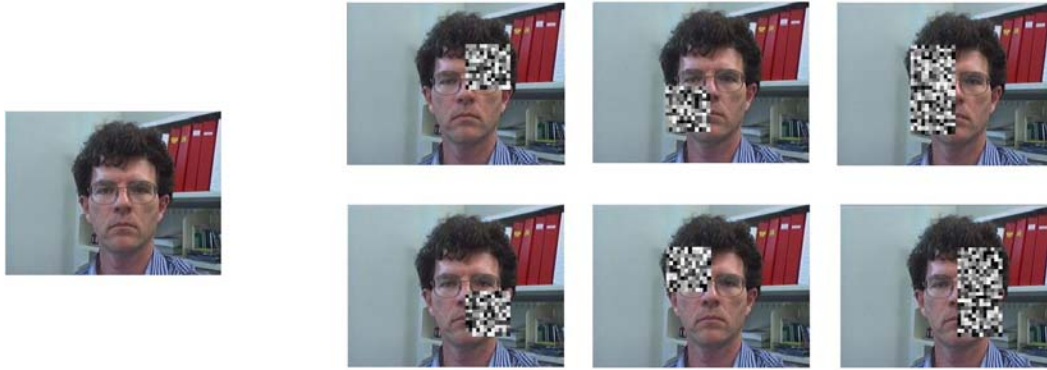


Figure 4.3: The occlusion process creates 6 additional positive examples for each frontal face.

6 different partial occlusions were calculated, varying from one quarter of the area of the sub-window up to half of the area. As the objective was to produce final 24x24 pixel images to the training process, each image was filled by either 12x12 or by 12x24 occlusion patches. All the patches' pixels varied from 0 to 255, randomly. The initial set of 1938 images composed a total of 13566 positive examples. Figure 4.3 shows an example of how the six additional positive images were created. It is important to stress that images from this person were not used in the training process, only images belonging to the FERET database were used as part of the positive set. The resulting classifier of the third training process was called *Classifier 3*.

The number of negative images stated in table 4.1 gives just an indication of how many negative images were used in each layer. The total number of negative images used depends on each layer's result. Initially, sub-windows with sizes of 24x24 pixels were used to train the first layer. As the training proceeds, new negative images are acquired from the same set by scanning the set and finding new sub-windows that are classified as positives.

4.3 Testing the classifiers

The sample classifier provided with OpenCV library worked very well for frontal faces, but it was unable to detect faces that presented any partial occlusion and shadows, especially if those were in the region of the eyes. The reasons for that were well understood, as the features with better weights tended to be around the eyes, cheeks and nose. Using only positive images without occlusions produced a classifier that disregarded images where occlusions could cause a significant change in the pattern. However, sometimes the partial occlusions coincided with the expected pattern and there was a consequent hit. It was very difficult to compare, appropriately, the performance of such classifiers because of the lack of standard occluded images from the literature. Certain parameters, such as scale

Table 4.2: Testing images: number of occluded frames and non-occluded frames.

	Total number of images	Images with partial occlusions	Images with faces free from occlusions
Test Set 1 (Akiyo)	90	0	90
Test Set 2 (web camera)	277	241	36

Table 4.3: Hit rates (%) for the sets of images using different classifiers.

	Sample	Classifier 1	Classifier 2	Classifier 3
Test Set 1 (Akiyo)	100.0	0.0	46.6	97.7
Test Set 2 (web camera)	57.0	44.0	44.4	74.0

factor and smoothing methods for scaling the sub-windows, might also affect the way each classifier was used. In all the experiments, a scale factor of 1.1 was used. In order to do an initial test, an image sequence known as Akiyo (Test Set 1) was used, but this sequence contained no partial occlusion. The sequence was used as a set of images independent from each other and the hit rate and false detection rates measured accordingly.

A test set was created using a web camera (Test Set 2), with a resolution of 640x480 pixels. Ideally, different people should be present in the set of images, but the experiments were limited to the scope of a simple face detection using one person on the test set. However, as all the positive images used in the training process did not present that particular person, it is reasonable to assume that the classifier would generalise in a similar way with any other faces. Table 4.2 shows how many frames, partial occlusions, and free faces that each set contains. Figure 4.1 shows examples of the partial occlusions that were created for test set 2.

4.4 Experimental results and discussion

The sample classifier from OpenCV worked very well with generic quasi-frontal faces, so it found 100% of the faces in the Akiyo sequence. As it was expected, the classifier 1 did not find any faces on the Akiyo sequence because it was trained with images belonging to a different person. Classifier 2 found most of the frontal faces, but the classifier missed most of the non-frontal faces where the angle of rotation was more than 15° . It is interesting to notice that the results improved dramatically on classifier 3, even though it was also trained with frontal occluded faces only. The random occlusion had an indirect effect of extending the range of the classifier.

The hit rates are shown in table 4.3. Classifier 1 and 2 are not as generic as the Sample Classifier, so the hit rates are about 44%. Classifier 3 shows an improved hit rate (74%), showing that the approach was successful.

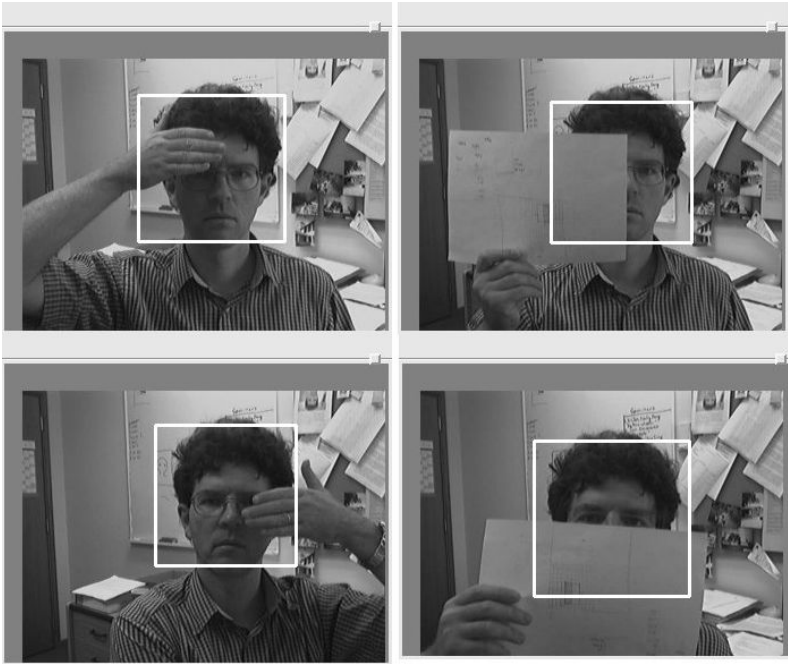


Figure 4.4: Hits in several occluded faces.

One problem that was faced by classifier 3 was the increase in the number of false positive detections. The reason was likely to be that the random part of the occlusions of the training examples caused Adaboost to choose features and weights that could be found in the background sub-windows.

Another possible cause for the high false positive detection rate in classifier 3 was that the number of the negative examples, kept at 1134 frames due to time and computational resources constrains, was too small. Despite the fact that the false detection was higher, when running it in real-time the problem would not be so critical because random false detection can be eliminated via a simple statistical method. The false detection tended to be randomly spread on the image. It was observed that even for two very similar frames, typical in video sequences, the classifier 3 yielded false detections in different locations of the images, while genuine hits were located on the region of the detected object. In a video sequence, the detections that reoccur in the vicinity of the previous one is more likely to be a hit, while detections that occurred once are more likely to be false detections. Figure 4.4 shows several occluded faces successfully found by Classifier 3.

A ROC curve was plotted to compare classifiers 2 and 3 (figure 4.5). The data for plotting the curve was obtained using the test set 2, which contained the partially occluded images. The ROC curve clearly shows that classifier 3, trained with the random patches over the image, has a better chance of finding the occluded faces.

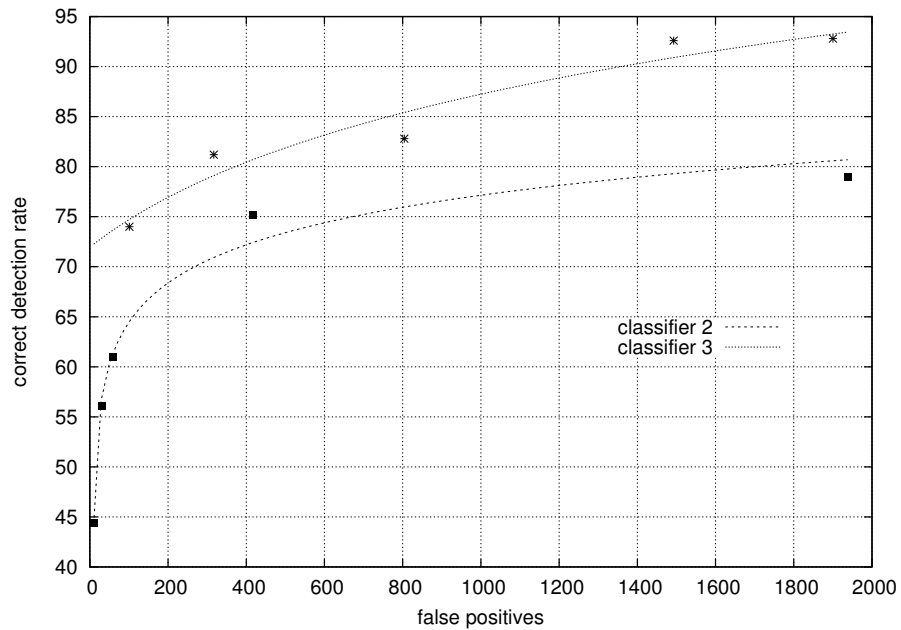


Figure 4.5: ROC curve comparing Classifiers 2 and 3.

4.5 Summary

In this chapter it was demonstrated that the classifier created by Adaboost using the OpenCV implementation could be robust to partial occlusions and shadow effects. The problem of getting good training sets was minimised using randomly generated patches to cover parts of the positive examples. The classifier created using this method yielded better hit rates when occlusions were present.

The positive example set can be increased via an automated process of adding random patches on selected areas. The training tends to concentrate on different parts of the kernel, producing a more generic classifier. The same method could be used to increase the number of positive samples by adding random background, as long as the training images can be easily segmented.

Chapter 5

Hand Detection

The recognition of hands is an important step towards gesture recognition and other applications related to human-computer interaction. This chapter presents experiments carried out with hand detection using Viola-Jones method. The experiments compared the performance of hand classifiers trained at different angles. The results showed that the Viola-Jones Method has the potential to be used for more sophisticated gesture and hand recognition.

The experiments carried out in this chapter shows the limitations for hand detection using the Viola-Jones method. Firstly, hand detection is more sensitive to rotation than faces. Secondly, square kernels are inadequate for hands due to the elongated shape of the hands (especially with certain gestures). Thirdly, multiple classifiers could be trained to deal with particular angles, but it costs both extra time to train and extra time to detect.

Despite the limitations of the experiments described in this chapter, they point to one important advantage of the method. All the classifiers, independent of angle or gesture to which they were trained, share the same SATs. Most of the classifiers looking at different angles do not need to compute the final layers of the cascade. This is an ideal situation to explore parallelism, which is discussed in chapter 9.

5.1 Related work

Many of the existing hand detector and gesture recognition systems rely on colour segmentation before further analysis (see for example Lockton and Fitzgibbon (2002)). Among the disadvantages of such methods is the fact they often are misled by other objects with similar colours (hands in front of the face is a typical situation that causes failure). The works of Ng and Ranganath (2002) and Stenger (2004) are good examples of hand detection based on colour segmentation.

Ng and Ranganath (2002) developed a system that could recognise the user's gestures and manipulate applications through a GUI. They have achieved more than 90% accuracy

processing up to 22 frames per second. The method used colour segmentation of the hand, extracting a “blob” from each image. The shape of the blobs were represented by Fourier descriptors and they were then input to a classification system. They investigated Hidden Markov models and recurrent neural networks as methods to achieve the final steps of gesture recognition.

Stenger (2004) developed a system to track hands based on a hierarchical Bayesian filter, integrating the information obtained from an initial single image. He successfully implemented a system with good accuracy, although it did not run in real-time. Other limitations were related to the use of Canny edge detection and colour segmentation, which caused problems when illumination conditions changed.

The use of Viola-Jones method for hand detection would potentially overcome the problem related to the use of colour segmentation. However, there are specific challenges in achieving good classifiers using the Viola-Jones method. A small rotation on the hand images cause a major change in the patterns. Training a single classifier for many angles did not seem to be a reliable option. Jones and Viola (2003) commented on the training of a single classifier to detect all poses of a face: “It appears that a monolithic approach, where a single classifier is trained to detect all poses of a face, is unlearnable with existing classifiers”. For hand detection this problem becomes more acute. Moreover, because many of the regions that contain the hand image would be oblong rather than square, using a single square kernel for training would include large portions of the background on training.

Ong and Bowden (2004) trained a tree classifier to detect hand shapes. They grouped similar hand shapes (gestures) via unsupervised clustering. Firstly, they trained a separate cascade for finding hands of any shape. Separate cascades were trained for the clustered sets. All the cascades were reunited in a tree like classifier. A total of about 2500 hand images was used to train the final classifier (composed by the various cascades). Another 2500 images were used in the test set. The detection errors were 0.2% for the hand detector and 2.6% (average) for the shape detector, surprisingly small for this type of detection. No ROC curve was plotted and no separate analysis was carried out for their experiments.

Kolsch and Turk (2004*a,b,c*) experimented with a modified version of Viola-Jones method for hand detection. They used extra Haar-like features, some with disjointed regions, arguing that the original feature set from Viola and Jones (2001*a*) was not discriminating enough for all hand appearances. In their experiments, they used their own test set collected from students images. A total of 2300 images were split in half for the training set and the test set. The ROC curves showed accuracies a little worse than what Viola and Jones found for face detection, which gives an indication about the generalisation of the method for other objects. They proceeded to show that the in-plane rotational sensitivity for hands was much more critical than that with face detectors. Hands rotated

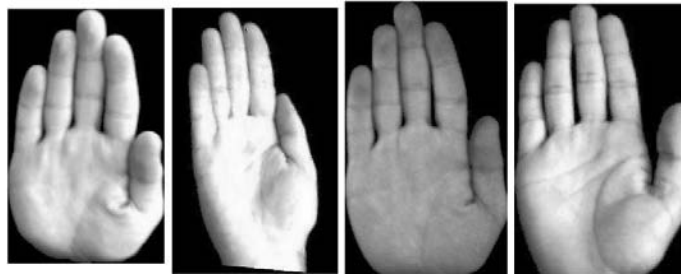


Figure 5.1: The basic hand images used on the training process.

by as little as 4° were very difficult to detect by classifiers trained at fixed angles. They then trained classifiers with examples rotated between 0° and 15° . They concluded that for examples within 5° of in-plane rotation, there was no significant loss of accuracy for their test set (Kolsch and Turk, 2004a).

The results of the work described above were limited in two aspects. Firstly, no separate analysis of the different angles of rotation was carried out. Secondly, they used square kernels, which includes a large portion of the background in the training, making it unclear how the background influenced the training process. The training and the test images presented similar background patterns, as the images for both sets were acquired directly from the same video sequences. An analysis of the choice of features during the training phase would be important in order to find out if parts of the backgrounds were being used.

5.2 Experiments with Haar-like features to recognise human hands

Every gesture made by a hand produces a different 2D pattern. Although there are hand image databases in the literature, (for example Athitsos and Sclaroff (2001) used a 3D model to create images for indexing and estimating pose), our own collection of images is used to have the flexibility and control of the samples. In order to analyse the performance and accuracy of classifiers created using Viola-Jones method, a simple gesture (shown in figure 5.1) was adopted. The hand images used by the training algorithm were acquired from 5 different individuals. The images were acquired under different illumination through artificial light, with a dark background. After a simple skin colour segmentation, the raw segmented images were used to prepare the positive set. The process of collecting examples is discussed in more details in Dadgostar and Barczak (2005).

5.2.1 Preparing the Positive Set

Based on the intensity and hue factor of the pixels with skin colour, an automated process segmented the hand in each image. Segmented hands were then added to a random

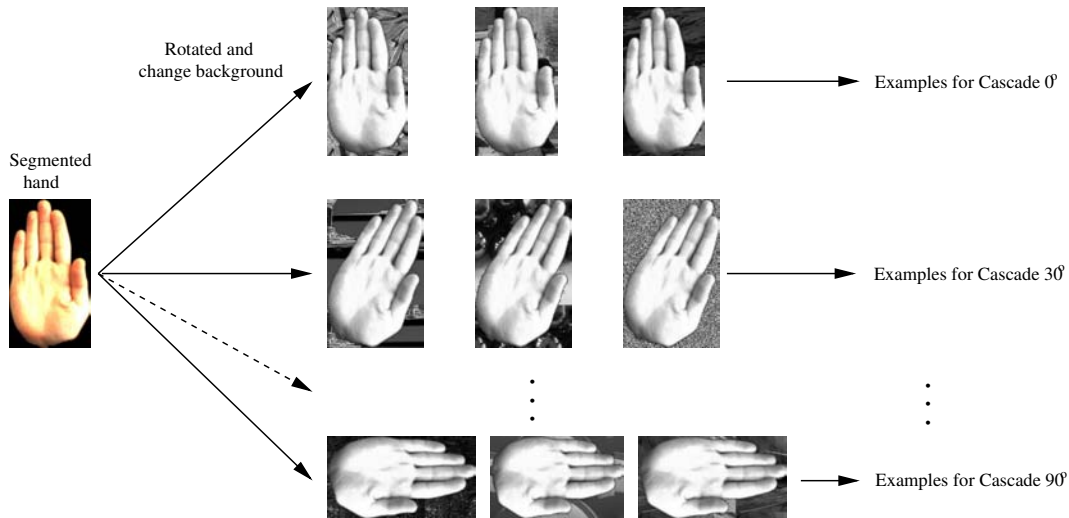


Figure 5.2: Creating the positive set.

background to facilitate the training procedure. Creating positive examples with varied background patterns was an important step, as this avoided the selection of Haar-like features located over the background. The creation process is shown in figure 5.2. An automated process was used to rotate the original set of images to angles from -90 to 90 degrees and introduce random backgrounds for each of the images. Each segmented image was reused 30 times for each angle. A total of 149 segmented images generated 4470 images with random backgrounds for each angle. A total of 19 orientations were used, at 12° intervals (from -84° to 84°) plus the angles 30° , -30° , 90° , and -90° .

5.2.2 Training

The cascades using algorithm 2, presented in section 2.2, were trained. Each cascade was capable of detecting hands (with the particular gesture of the samples) within a certain in-plane angle of rotation (figure 5.3). Some rotation tolerance was desirable because it was difficult to align the positive examples perfectly. The tolerance of the base examples were within 2° .

Instead of using an approach of converting cascades to produce another cascade at 90° (as done by Jones and Viola (2003)), training separately all the necessary cascades was the option adopted. The justification for that is based on the fact that different features might be chosen. It is not trivial to convert cascades from one angle to another when using tilted features, as well as oblong kernels, and the effects on the feature selection should be observed.

A modified version of Viola-Jones algorithm using the OpenCV (Bradski, 2000) library was used on training. 4470 images used to train each cascade, so a total of 84930 images were used (4470 images for 19 different angles). Different kernel sizes for each angle were used, in such a way that most of the area of the input was part of the hand. The kernel

Table 5.1: The kernel sizes for each cascade.

Angle	Kernel width (pixels)	Kernel height (pixels)
0°	24	42
12° and -12°	26	42
24° and -24°	30	42
30° and -30°	32	42
36° and -36°	36	42
48° and -48°	42	40
60° and -60°	42	32
72° and -72°	42	28
84° and -84°	42	24
90° and -90°	42	24

size for angle 0° was 24x42 pixels. The size was adjusted for each angle to keep the hands proportional until an angle of 90° , in which the kernel was 42x24 pixels. Table 5.1 shows a list of the kernel sizes for each angle.

5.2.3 Detection

Detection was carried out using a test set that was created with the same camera used to acquire the training set images. No colour segmentation was used for the test set. The images were converted to greyscale and had a resolution of 640x480 pixels. The cascades run concurrently and benefited from the fact that the same SATs were used for all cascades at any angle (figure 5.3). A modified version of the OpenCV detector was used to accommodate more than one cascade running concurrently.

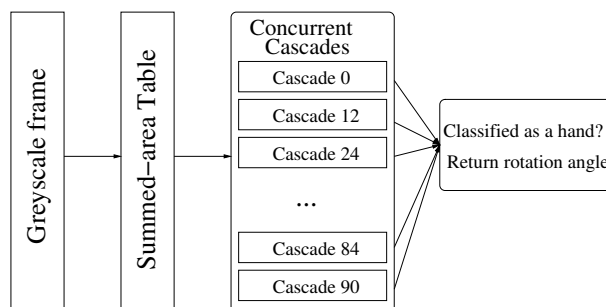


Figure 5.3: Using several cascades to recognise and determine rotation of hands

5.3 Results and Discussion

5.3.1 Accuracy

Two different test sets were used to individually measure the accuracy of the classifiers. Images for both were acquired by a web camera. The first test set was built by changing the hands position and then sorting the images by the approximation with the angle of the classifier (test set A, see image examples in figure 5.6). The second test set used similar images, but instead of sorting the images they were rotated to the appropriate angle. Images with similar illumination conditions were made available to assess the classifiers separately (test set B, see image examples in figure 5.7).

After analysing the choice of features, it was observed that selected features were adequately located on top of the hand region rather than on the background. Due to the discrete nature of the images, the rotation could not keep exactly the same patterns. The features were not always equivalent when training at different angles. Figure 5.4 shows the Haar-like features chosen by AdaBoost for the first layer of three cascades (angles 0° , 15° and 90° , respectively). A comparison of the cascades for these three angles showed that each training achieved different Haar-like feature targets (for each weak classifier).

Table 5.2 shows some of the results for the detection, considering that the minimum false detection was reached. The results showed an average hit rate of 62.7% hits, with a maximum of 84% and minimum of 34% detection. For some of the angles, the results were very poor, indicating the base images were not rich enough in lighting variation. The false positive rate on average was around 7.7%. The hit rates were lower and the false detection ratios higher than the ones reported in Kolsch and Turk (2004a) (for 95% hit about 0.001% of false detections) and Ong and Bowden (2004) (99.8% hit and 2.6% false detection). Both groups used images acquired under fixed conditions and split their image database into training set and test set. In these experiments, one could not do that, as the original images used to generate the training set had a constant background. Instead, images for the test set were acquired directly from a web camera.

Figure 5.5 shows the ROC curve for the seven classifiers of table 5.2. This curve was created by taking layers from each cascade (Viola and Jones (2004)). The scaling factor was 1.2 and the translation factor was 2 pixels. The tolerance for the detection rectangle position and size was set to 20%.

To assess and compare different cascades, a second performance measurement was created as follows. Images from the initial test set (780 images) were rotated back to an angle of 0° (see example in figure 5.7). The images were initially assessed using the cascade 0° . A set of 100 images was selected in such a way that the cascade 0° yielded 100% of hits and 0% of false detections. The images were then rotated again at the angle specified by each cascade. Table 5.3 shows the hit rate and the false positive for each

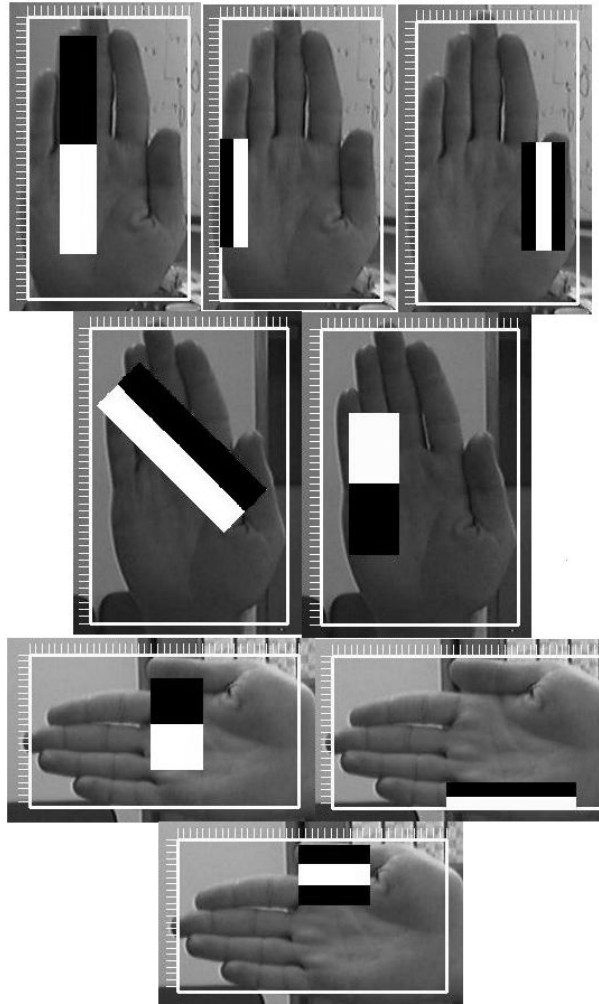


Figure 5.4: The Haar-like features chosen for the first step of each cascade.

cascade. The hit rate shows that the cascades are not completely equivalent, even though they were created using the same basic positive set. This happens due to the rounding process, both at the rotation and the computation of each Haar-like feature.

5.3.2 Performance

A set of cascades were used to estimate the maximum possible frame rate, and its relationship to the number of cascades being used concurrently. As expected, the speed of classification slowed down due to the extra computation, but the rate drop was not linear (Figure 5.8). Considering that the same SATs were used for all cascades, the reason behind this phenomenon has to be explained.

The rate was measured 10 times during a period of 5 minutes using a web camera on the desktop 1 (Pentium 2.4 GHz machine with 500MB memory running Linux 2.4). The results are shown in figure 5.8. The frame rate dropped significantly until about 8 cascades were used. Beyond this point the frame rate dropped more slowly. This

Table 5.2: Hit ratios and false positives for the first test set (test set A).

Angle	Number of Samples	Hits(%)	False Positives
-90	109	34	7
-60	104	62	9
-30	117	78	10
0	100	72	10
30	121	70	17
60	100	84	4
90	129	49	3
Total	780	63	60

Table 5.3: Comparing the cascades for different angles using a rotated test set (test set B).

Angle	Hits(%)	False Positives
-90	36	19
-84	85	21
-72	68	11
-60	79	47
-48	91	21
-36	91	16
-30	87	16
-24	77	54
-12	46	73
0	100	0
12	44	16
24	76	31
30	79	33
36	69	32
48	71	31
60	78	43
72	61	62
84	38	31
90	71	88

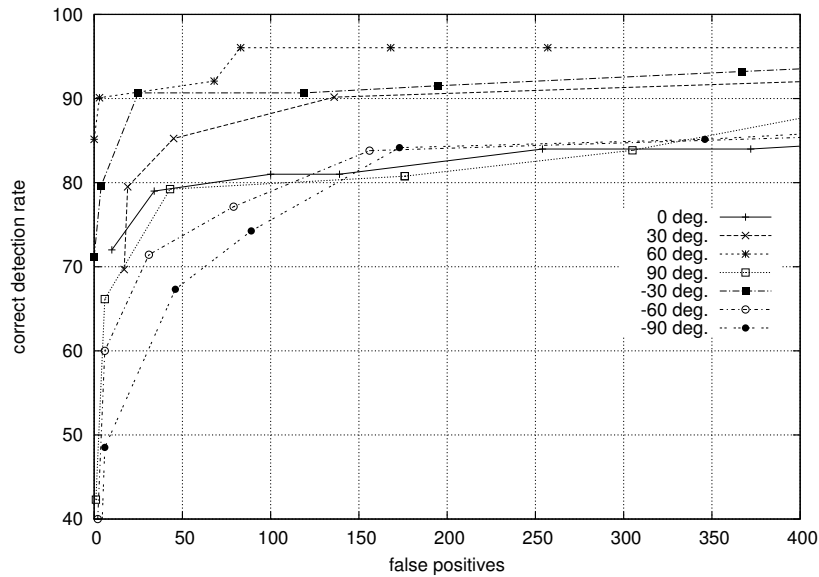


Figure 5.5: ROC curves comparing the classifiers trained at different angles for test set A.

somewhat unexpected behaviour has an explanation. As the sub-windows are tested by an increasing number of cascades, only a few cascades need to compute their last stages. Most cascades only compute the first few layers before eliminating all sub-windows. This shows a potential for exploring parallelism. Cascades that are not “active”, i.e., are not detecting any object in the early stages, use fewer processing resources than the ones that reach the final layers.

It is clear that a more efficient rotation approach would be necessary to cover a generic gesture recognition. For example, in order to recognise all ASL (American Sign Language) signs many different cascades at different angles would be needed. Apart from the problem of the long training time, the final set of cascades would run too slowly to be useful in real-time applications.

Figure 5.6 illustrates the application of the method. Notice that because the angle of detection was returned by the system, rectangles with appropriate sizes that encompassed the hand could be drawn.

5.4 Summary

In this chapter, several experiments with hand detection have been presented. The collection of positive examples was made efficient by generating random backgrounds. Experiments showed similar results for angular tolerance as in Kolsch and Turk (2004a). Aligned positive examples yielded better accuracy for a particular angle of rotation. Classifiers for different angles presented different accuracy despite using the same base hand images. The set of Haar-like features chosen by AdaBoost during training were different for each angle. Multiple cascades are more efficient at detection time than initially thought. Most

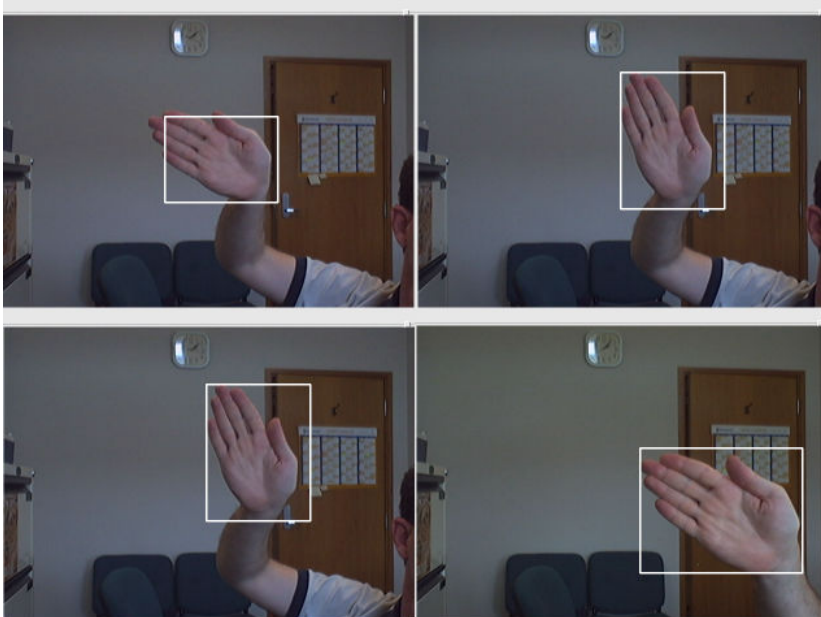


Figure 5.6: Examples of detection at different angles (from test set A).

of the cascades that classify hands at an angle use only a limited number of layers when compared to the cascade that finds the hand. Potentially, this strategy can be used for different gestures giving real-time performance.

The results presented in this chapter posed two more questions. Was it possible to convert an existing classifier to any angle? Jones and Viola (2003) suggested that this was the case for multiples of 90° in relation to the training samples' angle, but for other angles, no simple solution was presented. Detailed experiments regarding Haar-like feature extraction considering rotation are presented in chapter 6. The second question was: given the behaviour of multiple cascades, what would be the best parallelisation strategy for the detection stage? This question, as well as other related parallelisation issues, are discussed in chapter 9.



Figure 5.7: Examples of images in test set B

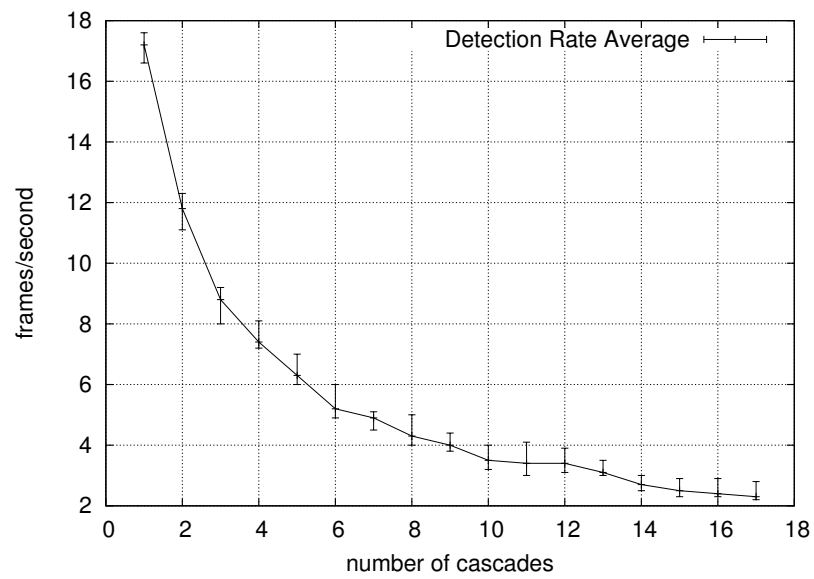


Figure 5.8: Rates for cascades detecting hands in various angles

Chapter 6

A Rotational-invariant Approach for Haar-like Features

This chapter proposes a new approach to detect rotated objects at distinct angles using the Viola-Jones detector. In this approach, the use of additional SATs makes an approximation of the Haar-like features for any given angle. The proposed approach uses different types of Haar-like features, including features that compute areas at 45° , 26.5° and 63.5° of rotation. Given a trained classifier (using upright features), a conversion is made using a pair of features. As a result, an equivalent feature value can be computed for any angle. This approach is called PEF (pair of equivalent features). The classifier's conversion is only an approximation and, therefore, one needs to know how the errors would affect the final accuracy of the classifier. The sources of errors in the computation of the Haar-like features are discussed, showing that for angles that are multiple of 45° the errors are often negligible. The main objective in this chapter is to use a single classifier (trained at a fixed angle) to achieve rotation invariance in such a way that the detection performance impact can be minimised. The questions that this chapter addresses are: is it accurate to convert classifiers for other angles? How to build SATs for angles different to 0° and 45° ?

In section 6.1, a background is presented, discussing four possible methods for Haar-like feature extraction under rotation. In section 6.2, the error sources for Haar-like features under rotation are discussed. In section 6.3, the proposed method for detection at generic angles is explained. Section 6.4 presents the results of four different experiments. The first two experiments assess the error of the feature extraction using the proposed approach. The last two experiments measure the accuracy of converted classifiers. Finally, section 6.5 summarises the limitations and possible improvements on a generic rotation invariant detector using Haar-like features.

6.1 Background

Due to the non-invariant nature of the Haar-like features, classifiers trained with this method are often incapable of finding rotated objects. It would be possible to use rotated positive examples during training, but such a monolithic approach often results in inaccurate classifiers (Jones and Viola, 2003). Besides, such a classifier would not be able to detect at which angle the object is found.

Jones and Viola (2003) experimented with rotated images for face tracking by training extra classifiers for various angles. Due to the angle tolerance yielded by face classifiers (about 15 degrees), they only needed to train 3 classifiers for 0, 30 and 60 degrees. The remaining classifiers for angles that are multiple of 30 could be created by either mirroring or rotating each feature by 90 degrees.

In practice, it is only possible to compute Haar-like features accurately at a fixed angle. The angle is determined by the way the SAT is created. Using the normal SAT, Jones and Viola (2003) suggested that if the feature type was converted, it would be also possible to compute features at other quadrants (90° , -90° and 180° from the original) using the same SAT. Therefore, one could compute features at eight different angles using two SATs. For certain applications, that would be enough. However, hands are an example of object that is much more sensitive to rotation, as seen in chapter 5. If one could convert existing classifiers to 45° using the tilted SAT (Lienhart and Maydt, 2002), an extra 8 angles would be added to the detection application without further training.

There are several possibilities to detect rotated objects, varying from brute force methods to parallel classifiers trained separately for a certain angle. Four possible methods are discussed next.

- **Method 1: brute force**

In a brute force method the entire image would be rotated at many angles. A single classifier would be applied to each resulting image. The implementation would be very simple and only one classifier per object would have to be trained. However, Rowley et al. (1998a) discussed this possibility and argued that the cost of rotating and repeating the classification process for each rotated image is computationally expensive. For example, Zhu et al. (2004) used a brute force approach by rotating the images at many angles before using a single cascade to detect faces. He found the method to be very simple, but slow. In the case of Haar-like features, there is a strong disadvantage in such an approach, as new SATs for every angle would have to be computed.

- **Method 2: Rowley's approach**

Rowley et al. (1998a) have proposed an algorithm for detecting rotated faces which he called "rotation invariant" face detection. They achieved good results using

a special algorithm to rotate face images before trying to recognise them. This approach required two classifiers based on NN. The first classifier yields an angle independent of the object contained in the sub-window. The second classifier is a normal face classifier, trained with aligned vertical face images. At the detection phase, the sub-window is rotated according to the angle given by the first classifier. The second classifier assesses the rotated sub-window. This approach worked better than a brute force approach, due to the fact that only the sub-windows that pass the first classifier have to be rotated.

The problem of using this idea for Haar-like features is that an angle finder would have to be trained specifically for several angles. In this case, training an angle finder would be as costly as training a standard classifier. A more critical issue is that the upright image SATs could not be used to compute the sub-window's Haar-like features. A new SAT would have to be computed for every sub-window, making the feature extraction process too slow.

- **Method 3: train all angles**

This method is slightly better than the brute force approach. This approach does not need explicit rotation of the image and the SATs can be computed only once for each frame. Training becomes an issue though, as many classifiers have to be trained independently. This method was used in chapter 5.

Li and Zhang (2004) used a mixed approach for face detection. Instead of training separate classifiers, they built a detector pyramid in which all the angles were examined at the top of the pyramid. At the second level of the face pyramid, the training set was split into three groups considering a limited range of angles. Finally, a third level split the training set into 9 groups. The final system has the capability of tracking in-plane rotated faces. Using the symmetry of the Haar-like features they would still need to train 8 separate classifiers.

- **Method 4: train a single classifier and convert it to other angles**

In this method, detection is guaranteed to be as fast as method 3, with the advantage that no extra training would be necessary. This would be the best compromise for both training and detection performances. The question is how to convert the classifiers appropriately. Jones and Viola (2003) suggested that for upright Haar-like features (types 0 to 7 of figure 6.1), it suffices to change the type for each weak classifier. For example, to convert a classifier to a 90° angle, a weak classifier with type 0 would be converted to type 1, type 4 would be converted to type 5 and so on (figure 6.1).

6.2 Error sources in the computation of Haar-like features

Because Haar-like features are scale invariant, computing a scaled version of the same image with scaled Haar-like features should yield the same value. The same can be said about rotating the image and computing the equivalent rotated features. In practice, when computing the Haar-like features in digital images, one can expect differences between the two computations. Although the features are a simple subtraction between areas, there are three main sources of errors:

- the approximations of the *feature sizes* due to scale changes.
- the approximations of the *feature positions* in relation to a fixed point in the image.
- the approximations of the *pixel values* due to scaling and/or rotating the image.

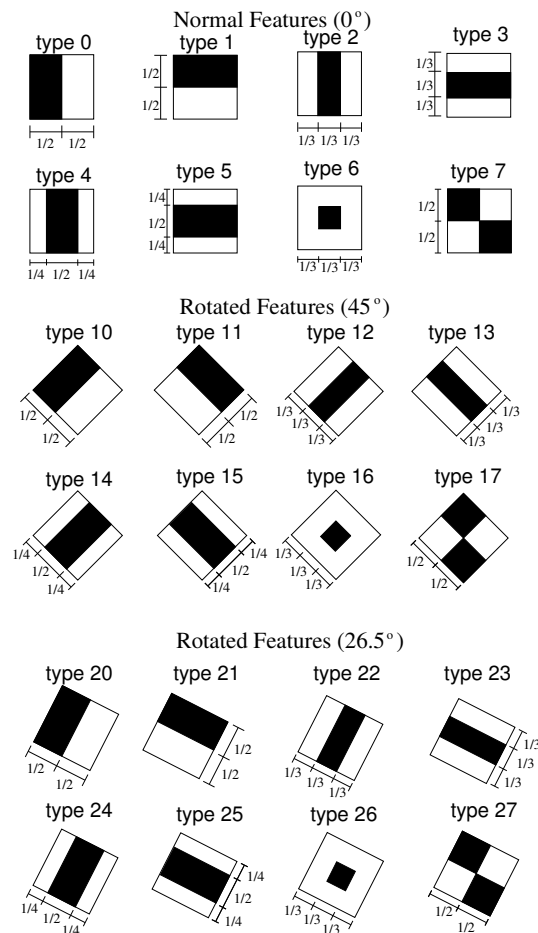


Figure 6.1: Classifiers using the Normal features can be converted to the other types.

The first error source can be partially compensated by a correction factor that ensures that the areas are proportional to the theoretical definition of that particular feature. Lienhart, Kuranov and Pisarevsky (2003) suggested a correction to this problem. A

correction factor is computed, so that the weights of the different rectangles of a feature keep the original area ratio between them. A simple test that verifies the correctness of the implementation is to find the value of the features over an image with constant pixel values. By definition (see equation 2.1), all features at any size and scale should yield zero for a constant image.

The second error source cannot be compensated without a more complicated approach such as computing sub-pixel values for the areas. This is not usually a good approach because the advantage of computing areas very rapidly with the assistance of the SATs would be lost.

The third error occurs after a processing operation such as scaling and rotation, when comparing features extracted from the original image with features extracted from its counterpart . Anti-aliasing techniques applied when scaling images can cause variations even if the features can fit the position and the sizes perfectly. Although this error is not directly related to the features' definition or their implementation, it is an important source of errors during the detection phase after classifiers are trained with a fixed scale.

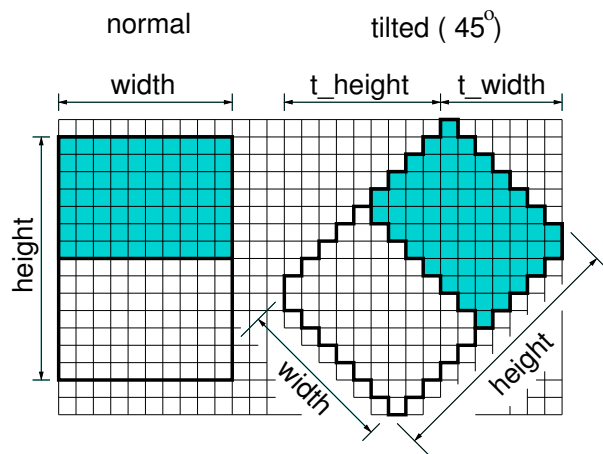


Figure 6.2: Comparison between a normal and a tilted feature (45° Haar-like feature).

6.2.1 Computing Haar-like features at 45°

Tilted Haar-like features were proposed by Lienhart and Maydt (2002) to test the hypotheses that a more robust classifier could be built if 45° features were included in the set. The normal features and the tilted features are not geometrically equivalent in digital processing, because the SAT used for the tilted features needs slightly distorted rectangles to correctly compute a similar sized area. With small resolutions, which is usually the case for the kernels used in Viola-Jones method, errors become more critical. Figure 6.2 shows an example for two equivalent features, one computed at 0° , the other computed at 45° . Notice the double pixel on two vertices of the tilted feature. This is necessary in

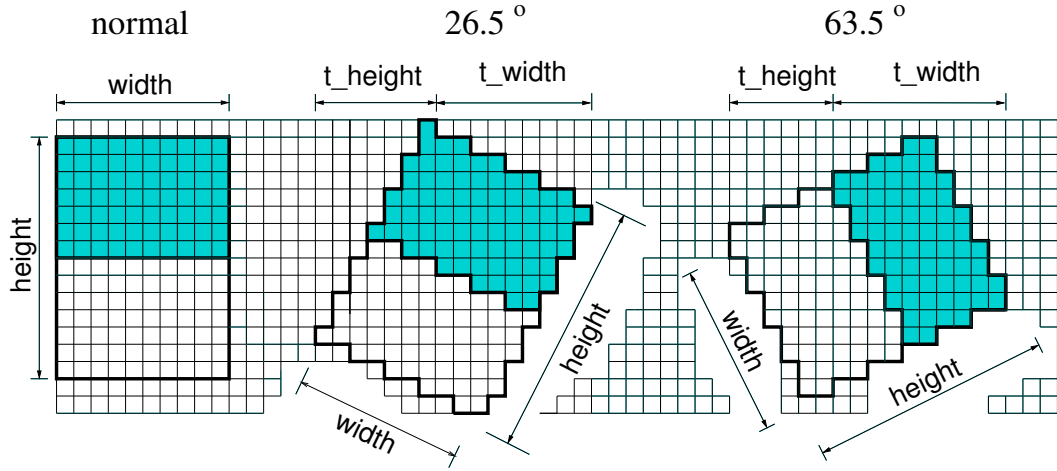


Figure 6.3: Comparison between a 0° , a 26.5° and a 63.5° Haar-like feature.

order to get the correct alignment of the pixels that belong to the feature (Lienhart and Maydt, 2002). Another approximation has to be made when computing the width and the height of the tilted feature, as this calculation often yields sub-pixel values.

6.2.2 Computing Haar-like features at 26.5° and 63.5°

Additional SATs would be able to compute other angles. Unfortunately, for most angles the solution is not trivial due to the pixelation caused by the rotation of a SAT. However, there is a special case where the ratio is 1:2 or 2:1. These two proportions are approximately equivalent to the angles 26.5° and 63.5° , which are near the angle that bisects the sectors between normal and tilted angles (exact angles would be 22.5° and 67.5°). As these angles also allow the computation on the other quadrants, four SATs would suffice to divide 360° in 16 regions (only approximately symmetric). Figure 6.3 shows an example for equivalent features computed at 0° , 26.5° and 63.5° .

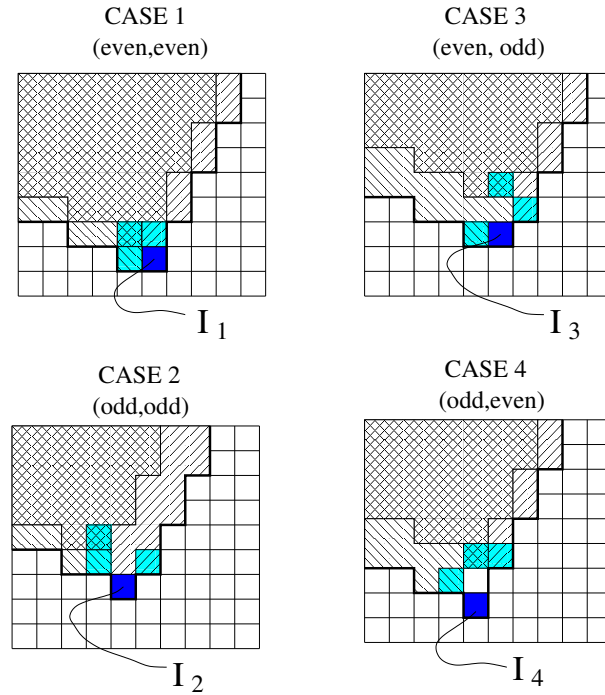
Next, the calculation of the SAT for the angle of 26.5° is formalised. The angle of 63.5° is analogous. Each element of the SAT covers different areas of the image. The covered area depends on the parity of the coordinates ((even,even), (even,odd), (odd,even) and (odd,odd) coordinates), shown in figure 6.4. The SAT can be computed recursively using the following equations:

$$I_{1(x,y)} = I_{(x-1,y)} + I_{(x,y-1)} - I_{(x-1,y-1)} + im(x,y) \quad (6.1)$$

$$I_{2(x,y)} = I_{(x+1,y-1)} + I_{(x-1,y-1)} - I_{(x-1,y-2)} + im(x,y) \quad (6.2)$$

$$I_{3(x,y)} = I_{(x-1,y)} + I_{(x+1,y-1)} - I_{(x,y-2)} + im(x,y) \quad (6.3)$$

$$I_{4(x,y)} = I_{(x-1,y-1)} + I_{(x+1,y-2)} - I_{(x,y-2)} + im(x,y) + im(x,y-1) \quad (6.4)$$

Figure 6.4: Computing the SAT for 26.56° recursively

Where: I_1 has an (even,even) coordinate, I_2 has (odd,odd), I_3 has (even,odd) and I_4 has (odd,even).

The computation of the Haar-like features at 26.5° also demands that the alignment of the 4 points of the feature are coherent with the creation of the SAT. Extra operations to analyse the parity of the coordinates are necessary. There are 64 possible combination of the parities of 4 points. Most cases do not need any correction. About 30 cases need a displacement of one position on one or two points, so the alignment is respected.¹

For example, in figure 6.5, an area is computed from a 26.5° SAT. Given an initial point $p_1 = (7, 7)$, a width $w = 6$ and a height $h = 3$, the other three points are:

$$p_2 = (x_{p_1} + w, y_{p_1} + \text{int}(\frac{w}{2})) = (13, 10)$$

$$p_3 = (x_{p_1} - \text{int}(\frac{h}{2}), y_{p_1} + h) = (6, 10)$$

$$p_4 = (x_{p_1} + w - \text{int}(\frac{h}{2}), y_{p_1} + h + \text{int}(\frac{w}{2})) = (12, 13)$$

Point p_4 is positioned in such a way that the area extracted from the SAT does not align with the area extracted for point p_2 . Equation 2.3 computes an area that adds a whole line of pixels, which should not be part of the feature's computation. By displacing point p_4 one pixel to the left, the alignment is correct.

This implementation used a unit SAT that counts the number of pixels included in a Haar-like feature. Although this requires extra lookups, it allows us to rapidly compute

¹Details of the implementation are described in the Appendix

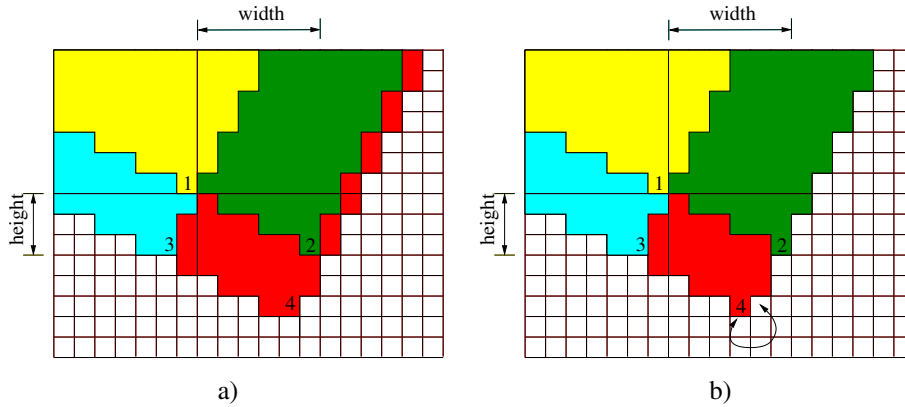


Figure 6.5: Corrections for SATs at 26.5° . a) the disalignment of point p_4 adds a line of pixels. b) By displacing point p_4 the alignment is correct.

the correction factors. One advantage of this approach is that the unit SAT does not change when new frames are being assessed. The unit SAT is only necessary for 26.5° and 63.5° rotated features. For the normal and tilted SATs, the number of pixels can be calculated directly using the width and height of the features (Lienhart and Maydt, 2002).

6.3 Converting features to generic angles

The extra SATs only allow Haar-like features at certain angles to be computed. In this section, it is proposed to use a function of the values of two features to approximate the value for a feature rotated at a generic angle. This approach is called *pair of equivalent features* (PEF). This is achieved using a weighted sum of the two equivalent features and a conversion of feature positions, feature sizes and feature types. Figure 6.6 shows how this conversion is done for the case where normal and 45° features are used. For an angle α , the new feature has to be positioned on a new kernel, larger in size to accommodate the rotation of the second feature.

For example, for angles between 0° and 45° , formula 6.5 applies. For angles larger than 45° there is also a feature type change to be made. For example, the PEF of a type 0 feature, for angles between 0° and 45° , is composed by a normal feature of type 0 and a tilted feature of type 10. If the angle is between 45° and 90° , the PEF is composed by a normal feature of type 1 and a tilted feature of type 10 (figure 6.7). For other angles, the PEF may be computed from other pairs, in some cases involving a change of sign.

The value of the PEF can be approximate by the following equation:

$$V = V_{normal} \cdot (45^\circ - \alpha) / 45^\circ + V_{tilted} \cdot \alpha / 45^\circ \quad (6.5)$$

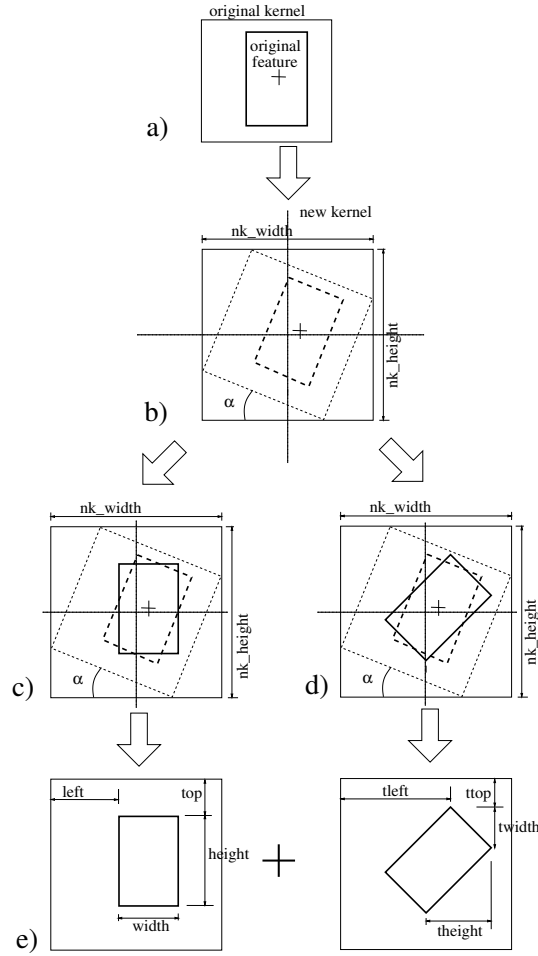


Figure 6.6: a) The original feature positioned in the original kernel. b) The new kernel size and the rotation angle are shown. c) The position for the new normal feature is computed. d) The size and the position is computed for the tilted feature. e) The resulting Pair of Equivalent Features (PEF).

Where: V_{normal} is the Value for the normal feature, V_{tilted} is the value for the tilted feature and V is the weighted average that depends on the angle α (between 0° and 45°).

Analogous to the case where the PEF is computed with a normal and a tilted SAT, one can approximate the feature value for any angle between 0° and 26.5° :

$$V = V_{normal} \cdot (26.5^\circ - \alpha) / 26.5^\circ + V_{26.5} \cdot \alpha / 26.5^\circ \quad (6.6)$$

Where: V_{normal} is the Value for the normal feature, V_{26} is the value for the feature at 26.5° and V is the weighted average that depends on the angle α (between 0° and 26.5°).

6.3.1 Error Analysis for Pair of Equivalent Features (PEF)

A Haar-like feature value depends on the distribution of the pixels in the area where it is applied. The maximum value for many types of Haar-like features occurs when an edge

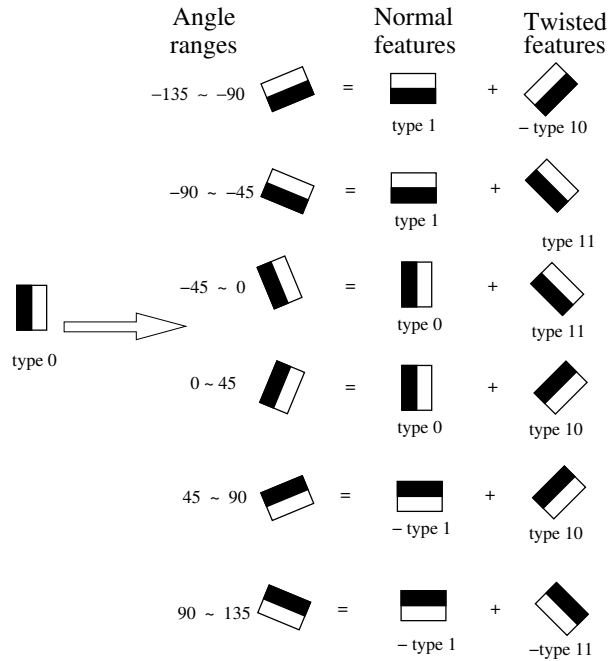


Figure 6.7: An example of PEF for a type 0 feature considering a number of angle ranges.

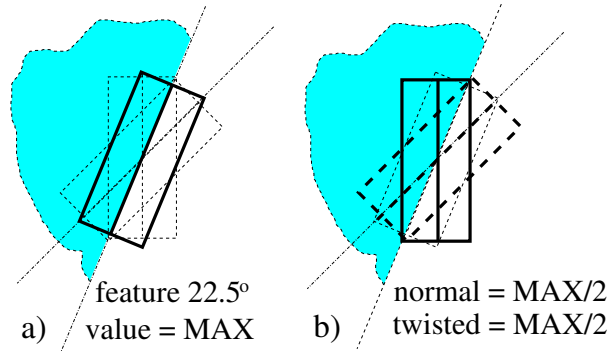
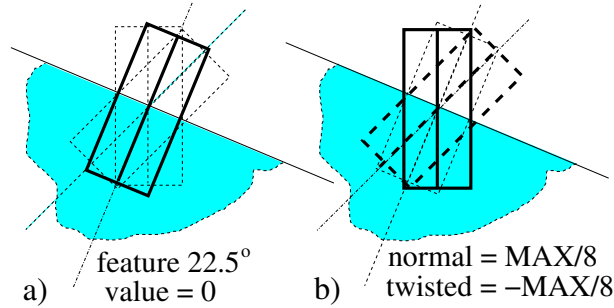
is located in the middle of the feature. In order to maximise the absolute value of the feature, it is also necessary that all the values of the pixels on one side of the edge are zeros and all the other pixel values are maximum (for example, 255 in a grey scale image).

From this point, the maximum value of a feature, at a generic scale, is referred to as MAX. Any feature value can vary from -MAX to MAX. Suppose that a type 0 Haar-like feature needs to be computed at an angle of 22.5° , over an edge presented by the image. The ratio width/height in this example is assumed to be approximately 2:5. The theoretical value for the figure 6.8-a is MAX. The PEF value is $\text{abs}(\text{MAX}/2)$ (figure 6.8-b), as both the normal feature and the tilted feature yield the same value MAX/2. For figure 6.9-a the value of the feature would be zero. For the normal features it is MAX/8 and for the tilted features it is -MAX/8 (figure 6.9-b). Therefore, the PEF value is zero, which is the required result. In the extreme cases an error of about 50% in relation to MAX is expected. Variations on these errors might be expected for different width/height ratios.

6.4 Experimental Results and Discussion

6.4.1 Experiment 1: error analysis for PEFs using 0° and 45°

In order to assess the impact of the approximation, an experiment was carried out using several natural images as well as binary images where the edges would call for the maximum value of the features. The images were rotated to angle α . Features at all possible

Figure 6.8: Case where the value of the 22.5° feature should be MAX .Figure 6.9: Case where the value of the 22.5° feature should be zero.

scales and positions where computed in the original image (figure 6.10). The PEFs for each of those features were also computed. Figure 6.11 shows the error measurement approach for the first frame of the Akiyo sequence at 45° . For each angle of rotation several thousand PEFs were computed this way.

The errors were calculated as a percentage of the maximum possible value (MAX) for that feature type at that scale (F is the original feature value and V is the PEF value for that angle):

$$error = \frac{|F - V|}{MAX} \quad (6.7)$$

The results for two sample images are presented: the first frame of the Akiyo sequence and a Chessboard (figure 6.12). The first image is significant because it contains a face, a common object used in detection algorithms. The second image yields large errors because it contains well defined vertical and horizontal edges. A small deviation on the position of the converted features may cause relatively large errors.

The original feature size used in the experiment was 20×20 pixels. The kernel size for the PEF was 34×34 pixels. Figures 6.13, 6.14, 6.15 and 6.16 show the maximum errors for both images using different type of Haar-like features. As expected, the largest errors were around the region of 22.5° . The errors at the angle of 45° decreased almost to the

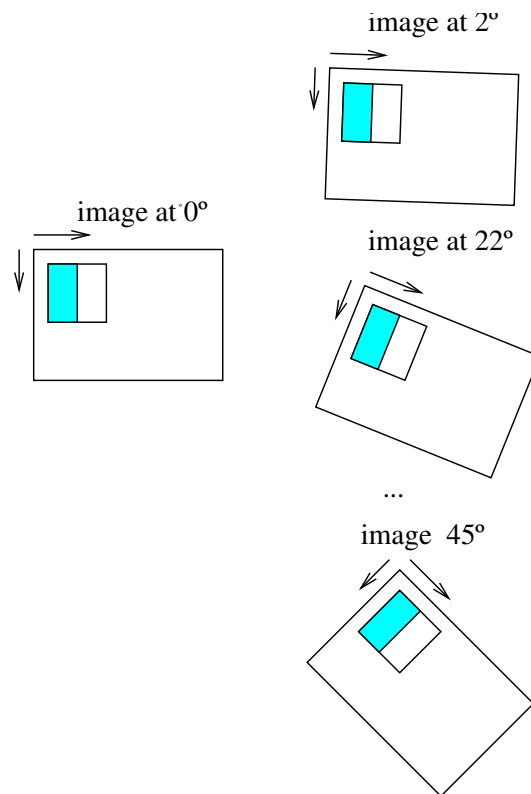


Figure 6.10: Scanning the images at various angles and comparing them to the PEF value.

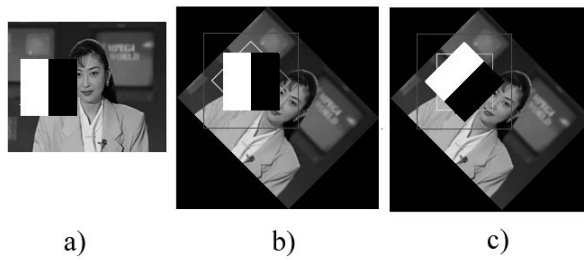


Figure 6.11: Measuring errors for PEFs. a) the original normal feature b) the converted normal feature c) the converted 45° feature.



Figure 6.12: First frame of Akiyo sequence and the chessboard images.

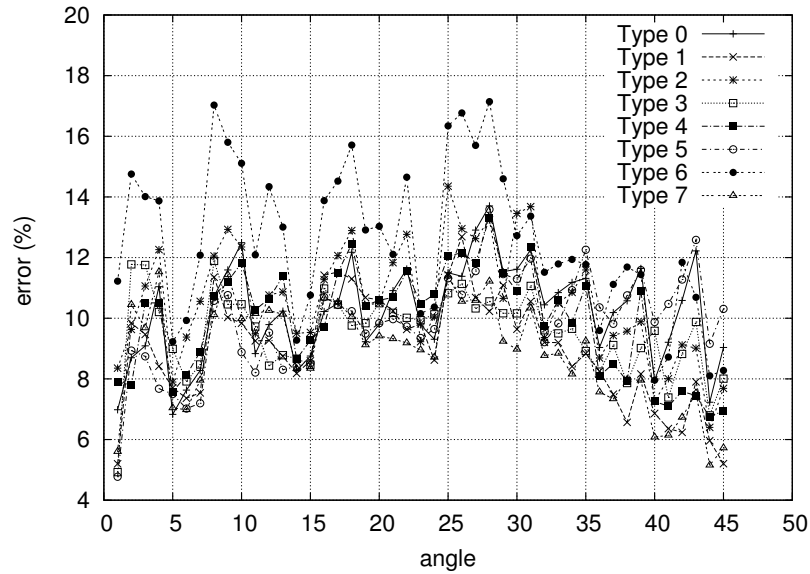


Figure 6.13: Maximum error vs angle for Akiyo (normal+45° PEFs).

same levels of those at 1°. This indicated that the tilted features could be used to convert normal features to 45° with good accuracy.

The maximum error for Akiyo was 17% for type 6 feature (figure 6.13). For the chessboard image the maximum error was 47% for type 7 features(figure 6.15). The maximum error in this case almost reached the theoretical value of 50%. The average errors, based on the absolute value of the PEFs, indicate that for most features the PEF values are actually very close to the original feature. For Akiyo, the maximum average error was around 1.1% for feature type 2 (figure 6.14). For the chessboard the maximum average error was 6% for type 7 features (figure 6.16).

6.4.2 Experiment 2: error analysis for PEFs using 0° and 26.5°

In this experiment, the third group of features on figure 6.1 was implemented. Figures 6.17 and 6.18 show the maximum errors obtained for Akiyo and for Chessboard. Figures 6.19 and 6.20 show the average errors.

The maximum error for Akiyo was 12% for type 3 feature (figure 6.17). For the chessboard image the maximum error was 34% for type 3 features (figure 6.18). The maximum errors were close to those obtained in experiment 1. For Akiyo, the maximum average error was around 1.4% for feature type 1 (figure 6.19). For the chessboard the maximum average error was 6% for type 7 features (figure 6.20).

It is clear that the errors at angles around 26° should be much smaller. The origin of these errors are related to the position calculations during the feature conversion. The positioning of the equivalent features at these angles are much more sensitive than the ones

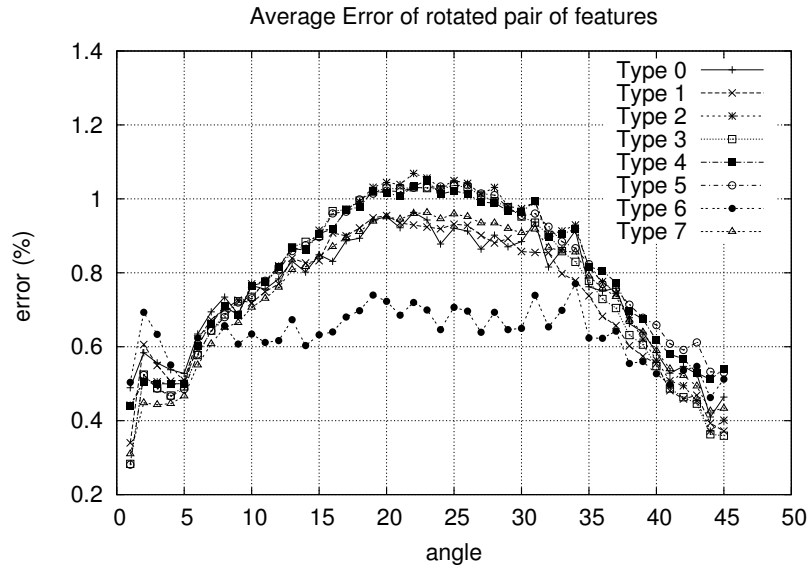


Figure 6.14: Average error vs angle for Akiyo (normal+45° PEFs).

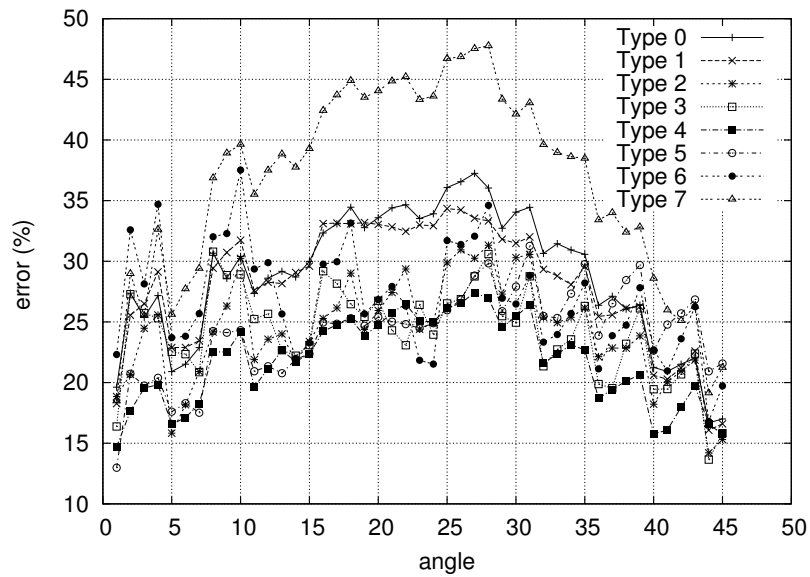


Figure 6.15: Maximum error vs angle for Chessboard (normal+45° PEFs).

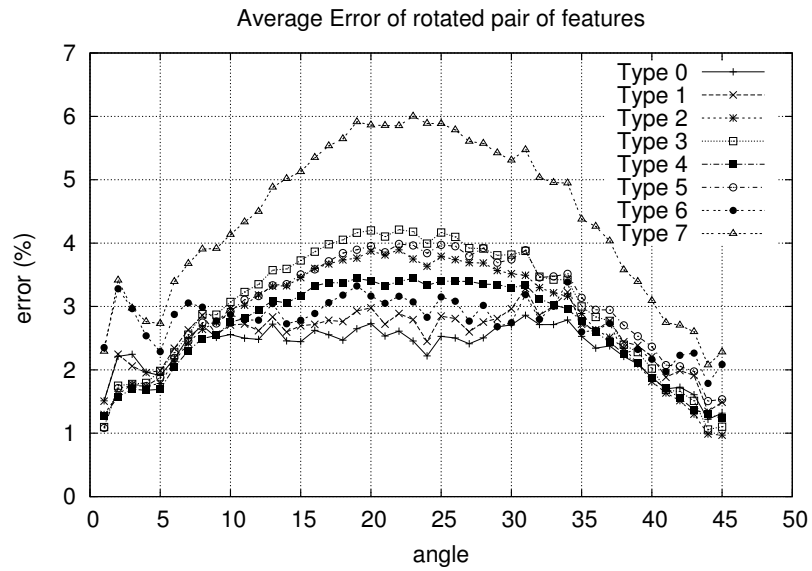


Figure 6.16: Average error vs angle for Chessboard (normal+45° PEFs).

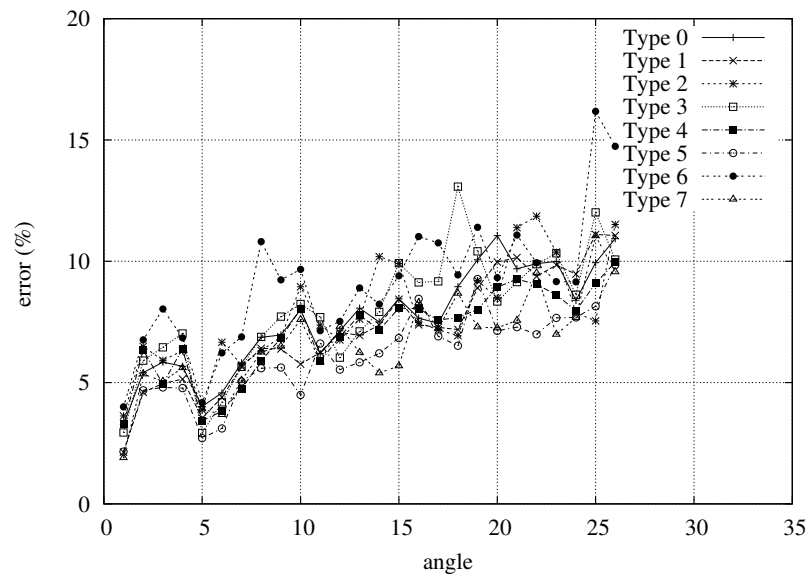


Figure 6.17: Maximum error vs angle for Akiyo (normal+26.5° PEFs).

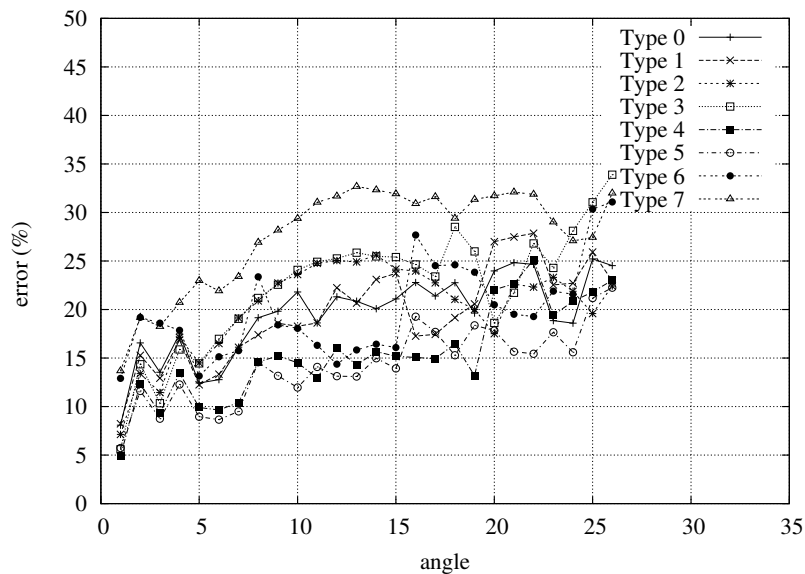


Figure 6.18: Maximum error vs angle for Chessboard (normal+26.5° PEFs).

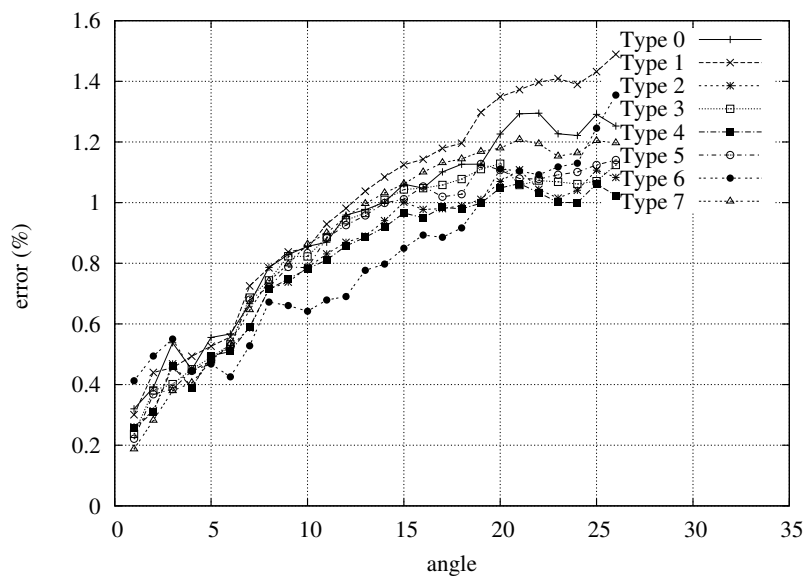


Figure 6.19: Average error vs angle for Akiyo (normal+26.5° PEFs).

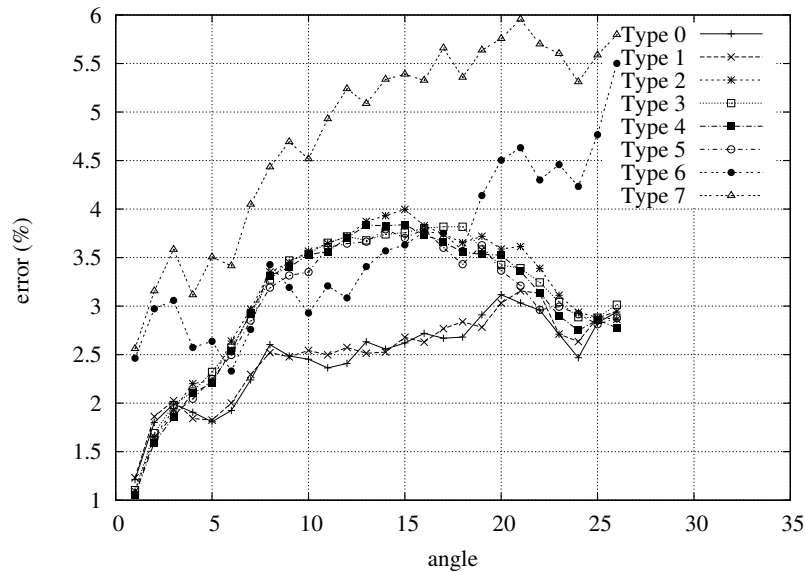


Figure 6.20: Average error vs angle for Chessboard (normal+26.5° PEFs).

in experiment 1. Any small deviation on one of the points in the SAT result in larger errors. The solution for this problem consists in improving the rounding algorithm to consider the best point depending on the parameters of the area being computed (position, scaling and parity of the coordinates).

6.4.3 Experiment 3: using PEFs to compute features in classifiers

Several classifiers were trained using 1000 frontal faces from FERET and reserving the other 500 frontal faces for the test set. These classifiers were tested and their results were compared at different angles. The procedure was to rotate the test set and try to detect the faces at that angle using PEF (normal+45°). The classifiers use kernels of different sizes (20x20, 24x24, 30x30 and 36x36), each with 16 layers.

Figure 6.21 shows the hit ratio for several angles. Notice that for the 45° the results are comparable to those at 0°, indicating that at these angles, the equivalence of the features are accurate. However, for angles at the vicinity of 22.5° the hit rates were very poor, specially for the small kernels.

More details on the 36x36 kernel, which presented results good enough to be used in practice, are shown. Figure 6.22 and figure 6.23 show the variation of the hit rate and false detection rates for this kernel size. A hit is considered when it is within a 10% tolerance area around the face. The false detection was computed conservatively as the total number of hits outside the tolerance area. On the worst part of the curve (at 22.5°), the 16 layers yielded 55% and the 12 layers 75% hit rate. The false detection was much higher for angles close to 0° or 45° than it was for 22.5° region. This indicates that a

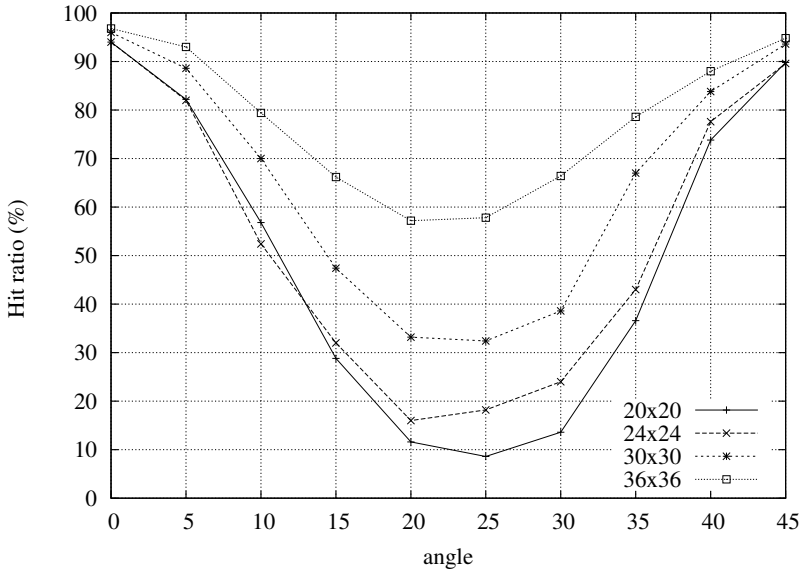


Figure 6.21: Hit ratio vs angle using 16 layers.

correction factor could be applied to force the classifier to work under similar conditions for all angles (same hit rates and false detection rates).

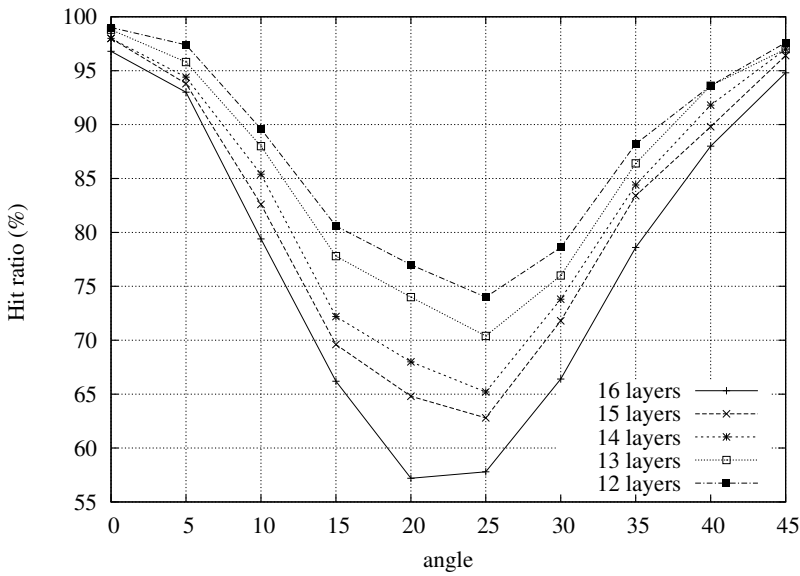


Figure 6.22: Hit ratio vs angle, kernel 36x36.

A correction factor can be applied in two ways. One option is to apply a correction factor directly to the computation of each PEF, because the value of a PEF is typically lower than the true value of the equivalent rotated feature. However, this implies finding different correction factors for each feature type. The second option is to apply a correction factor for the final threshold of each layer. If the correction factor is a function of the angle, the false detection levels can be adjusted properly. Based on the shape of the hit

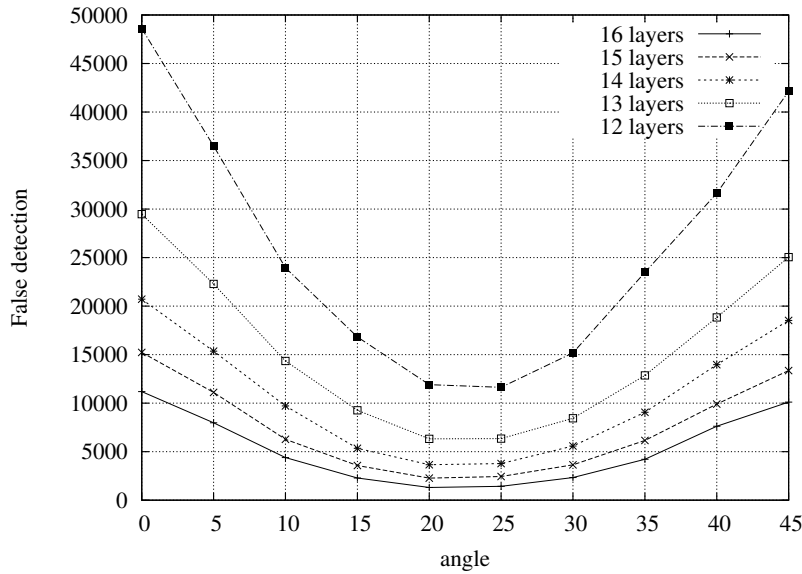


Figure 6.23: False detection rate vs angle, kernel 36x36.

rate curve, it was empirically found that the following formula is suitable:

$$factor(\alpha) = \frac{(1 - fc) \cdot \cos(8\alpha) + (1 + fc)}{2} \quad (6.8)$$

Where: fc is a constant between 0.3 and 0.9. Notice that at 0° or 45° the correction factor is 1 for any fc .

The results of the application of the correction factor can be seen in figure 6.24. Notice that an fc of 0.5 lowered the threshold by too much, compromising the false detection rate. To achieve a balance between the false detections and hit rates an fc of 0.65 was the best value. An alternative to a correction factor would be to use a different number of layers for each angle.

According to figures 6.8 and 6.9, longer features makes their PEFs prone to larger errors. In order to test this hypothesis, classifiers using only shorter Haar-like features were trained. The same set of features (figure 6.1) were used, but the ratios width/height and height/width were limited to 20:8. The results can be seen in figure 6.25. Comparing with figure 6.22, the results were very similar, indicating that even at that ratio the errors are too large at angles in the vicinity of 22.5° .

Using a web camera, the detection with PEFs was slower than using a single cascade with the original OpenCV version. This was expected because more Haar-like features need to be computed when using the PEF. An average of 3 frames per second was achieved, with a 640x480 pixels frame and a scaling factor of 1.2 and a translation factor of 2. Figure 6.26 shows a few frames with their hit rectangles and a few false detections. This application uses a simple criteria to choose the best hit. Each stage passes with a certain

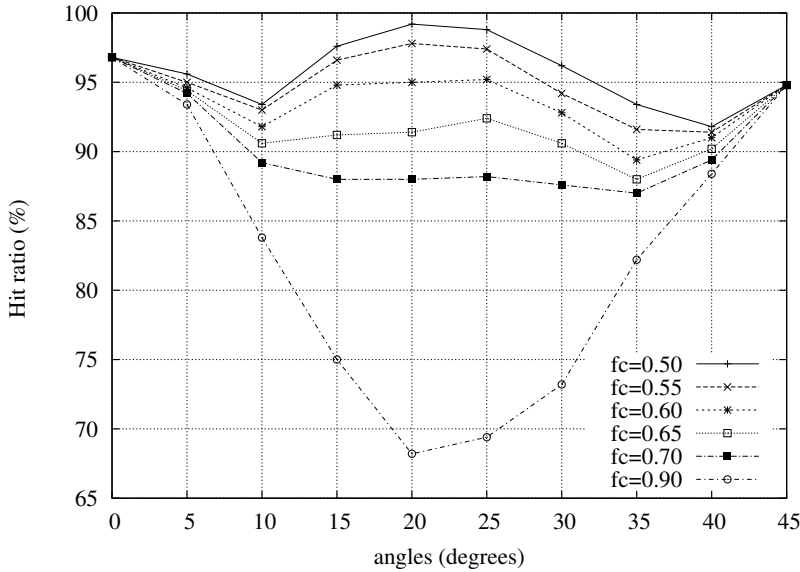


Figure 6.24: The classifier 36x36 with various correction factors.

sum of weak classifiers. The hit with the largest sum is chosen as the best choice.

The performance of the detector at a particular angle was impacted by the fact that twice as many features as the original classifier have to be computed. One could argue in favour of keeping the first few layers in memory, so the computation could be shared among several classifiers to minimise the impact in performance. On the final layers, there should be very few candidate sub-windows that are classified, so this form of buffering would not be necessary.

6.4.4 Experiment 4: converting the original face classifier using tilted features

In order to try to improve the Haar-like feature computation errors, one last experiment has been done. Rather than convert the classifier to 45° using a kernel of a limited size as before, a larger kernel was used (compare figures 6.2 and 6.27). This allowed for a more accurate conversion process. The size of the new converted kernel was large enough to accommodate all Haar-like features used by the classifier, keeping the same proportion without rounding (in both sizes and distances of the features). The standard frontal face classifier available in OpenCV was used for this experiment. The original OpenCV classifier was trained with a kernel size of 24x24 pixels. The converted tilted classifiers were converted to a kernel size of 48x48 pixels. The cascades converted directly from the upright cascade were used to assess the hit rates and false detection rates. In order to keep the same conditions and use the lighting contrast correction used in OpenCV, an extra SAT had to be implemented to return values for squared pixels of the tilted Haar-like features.

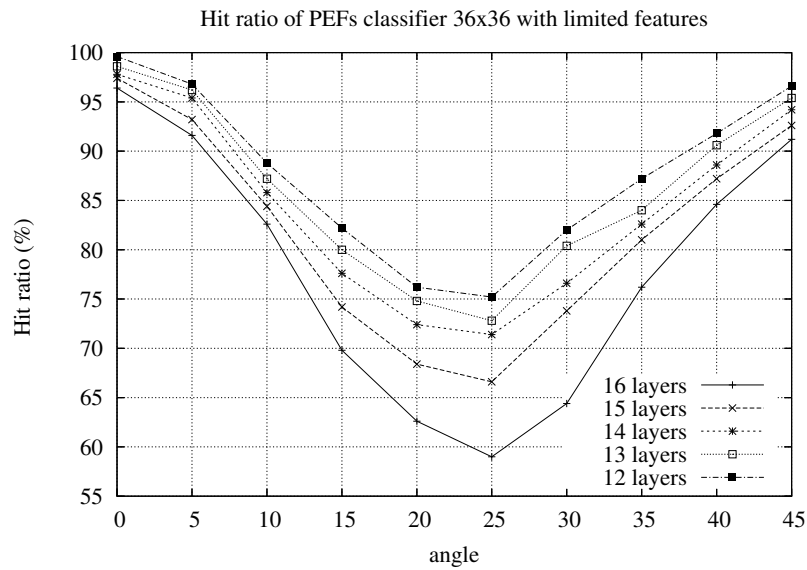


Figure 6.25: The hit rate for classifier 36x36 with limited range of features.



Figure 6.26: Detecting rotated faces using PEFs. The white rectangle indicates the best match. The green rectangle is the kernel for the best match. The red rectangles represent other detections (including false detection).

The accuracy comparison was made using a modified CMU-MIT database. Each image of the CMU database was rotated by 45° or -45° . Because many of the faces presented in the database are smaller than the kernel size for the tilted cascades (48x48 pixels), some of the images had to be scaled up. Figure 6.28 shows the ROC curve comparing the upright cascade and the converted tilted cascades. The results show that the converted cascades were slightly less accurate than the upright cascade for the CMU-MIT database. The results were collected with a scale factor of 1.2 and a translation factor of 2 pixels. The fact that the images were scaled up made the occurrence of false detections more common for the tilted classifiers.

Despite the drop in accuracy for the CMU-MIT database, the detection was very good when using the web camera with real faces. Figure 6.29 shows an example of the system working with three concurrent cascades, at 0° , 45° and -45° using the web camera.

The performance of the system using more than one cascade was expected to be slower. Table 6.1 shows the results for various combinations of cascades. The rates were

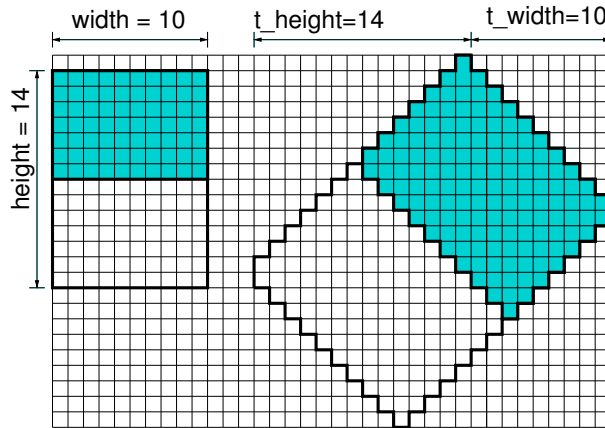


Figure 6.27: Converting Haar-like feature maintaining the proportional area

Table 6.1: Performance of tilted cascades.

Cascades	rate (frames/second)
normal	10.1
tilted	9.0
2x tilted	6.8
normal + 1 tilted	7.9
normal + 2 tilted	5.7

measured for one minute in a Dual Core AMD Opteron(tm) Processor 170 2.0 GHz desktop computer. The performance drop was similar to the results collected for figure 5.8 when using three concurrent hand detection cascades.

The main advantage of using the scaled tilted features was that the features kept the same proportions and positions in relation to the kernel. Long features did not have the drawback of the PEF approach. The only disadvantage of making the kernel larger is that the classifier is not able to detect sub-windows as small as the up-right classifier.

6.5 Summary

In this chapter, a new approach for a rotational invariant Viola-Jones detector was developed. The proposed method converts a previously trained classifier to work at any angle, so rotated objects are detected without specifically training the classifier for that angle. In order to establish the accuracy of the converting method, the errors associated with each feature type and with rotation were analysed.

The first experiment showed that tilted features can successfully convert a normal feature to angles of 45° . The PEFs suffer from large errors that affects the accuracy of the classifier for angles in the vicinity of 22.5° .

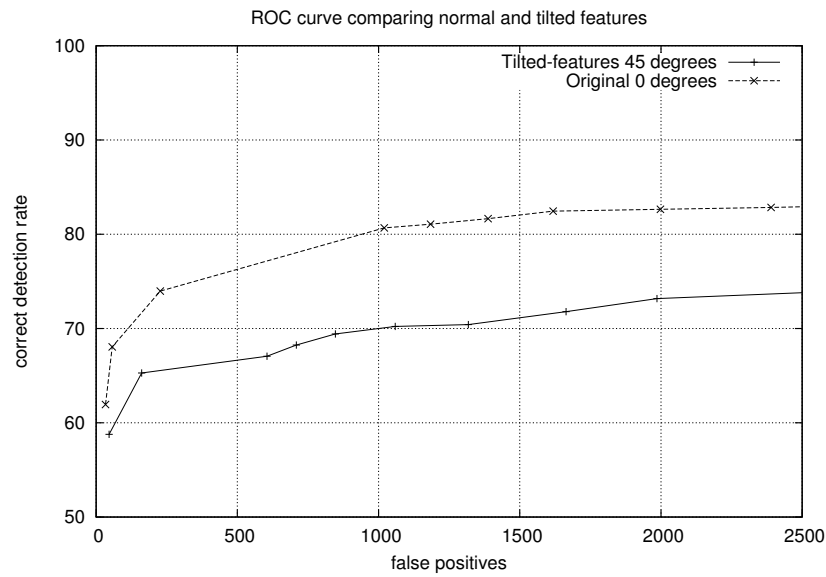


Figure 6.28: Comparative ROC curve for upright and tilted cascades for the CMU-MIT database.



Figure 6.29: Some results for the tilted cascade converted from OpenCV.

The second experiment showed that the implementation of the SAT for angles of 26.5° needs improvements to successfully compute equivalent features at angles of 26.5° . The errors are too large for this approach to be practical.

The third experiment, based on classifiers using the PEF approach for two angles (0° and 45°), showed accuracy limitations when working at angles in the vicinity of 22.5° . A correction factor is used to improve the hit rates. Also, this experiment showed that the errors of the conversion are minimised when using larger kernels.

The fourth experiment showed that by using a large kernel for the 45° cascade, better results for converting classifiers can be achieved. The OpenCV sample classifier for frontal faces was successfully converted to angles multiples of 45° . The accuracy assessed with the CMU-MIT test set is acceptable and comparable to the original upright OpenCV sample cascade. Moreover, tests with a web camera shows a very robust classification at angles 45° and -45° .

The limitations of Haar-like features pose the following question: is it possible to use

rotational invariant features to train cascades with AdaBoost? Features such as moment invariants are rotation invariant, but their accuracy, performance and discrimination powers need to be tested. These issues are discussed in the next chapter.

Chapter 7

A Real-time Approach for Moment Invariants

In this chapter, a new feature extraction method is proposed. The new method combines two well known methods, SAT (Crow, 1984) and moment invariants (Hu, 1962), and incorporates the knowledge about independence of moments studied by Flusser (2000*a*). The method allows the use of geometric moment invariants in real-time applications, as it can compute features very rapidly.

In section 7.2 the computation of moments using SATs is discussed. The method of feature extraction is extended to include contrast stretching, based on what Lienhart and Maydt (2002) did for Haar-like features. The rapid extraction method with contrast stretching minimises the effects of changeable lighting conditions.

In section 7.3 an original feature extraction method is discussed. A new feature extraction method that uses moments computed over several concentric discs (called CDMI) is proposed, extending the limited number of independent moment invariants. An approximation of discs allows the generation of geometric moments over circular areas rather than rectangular ones (see figure 7.6). The accuracy of the feature set produced by this method is analysed under rotation and scaling.

Finally, in section 7.4 the new proposed feature extraction method is used to train classifiers for face detection. The first experiment used rectangular areas and 11 moments per sub-window and shows that classifiers are limited in accuracy. The second experiment used concentric discs to compute 66 moments per sub-window. CDMIs worked successfully in face detection with images acquired by a web camera, despite the limitations in the training process due to the weak discriminative powers of moment invariants. The third experiment used concentric discs to compute different numbers of moments per sub-window, showing that the use of concentric discs improved the training process in both speed and accuracy.

Based on an extensive literature review, we can state that this is the first time that

the moments invariant and the SATs are combined to compute moments of concentric discs, as proposed in this work.

7.1 Related work

Hu's moment invariants were first proposed in 1962 (Hu, 1962) and are still widely used today for computer vision algorithms, although other invariant features have been proposed since. Reiss (1991) found some minor errors in Hu's theory regarding the invariant properties to general linear transformations and proposed corrections to deduce three other moments that are also invariant to illumination. Hu's original equations are invariant to translation, mirroring, scaling and most importantly for this work, to rotation.

It is interesting that it took almost 40 years between the publication of the moments by Hu and the realisation that some of Hu's moments were dependent. In the last few years, there has been a renewed interest in the moment invariants theory proposed by Hu (1962). Flusser (2000a) discussed the independence and completeness of the original Hu's set (see also Flusser (2002)). Two of the seven moments proposed by Hu were dependent on the others, leaving only five to effectively be used for classification of images. Moreover, Flusser developed a method to find out the best sets of moments for higher orders. Flusser proposed six independent moments (of 2^{nd} and 3^{rd} orders), of which five were equivalent to the original Hu's moments and one was new. Flusser and Suk (1993) studied invariants based on moments when they proposed a set of four affine moment invariants (2^{nd} and 3^{rd} order). Suk and Flusser (2003) extended the theory to incorporate blurring and affine invariance.

Fast algorithms for the computation of moments can be grouped into two classes (Flusser, 2000b). One group proposed algorithms based on the decomposition of images into row segments (e.g. Flusser (1998)). The other group of methods is based on Green's theorem (see Yang and Albrechtsen (1996) for a review), which uses the object's boundary to compute the moments. Spiliotis and Mertzios (1998) proposed a real-time algorithm to compute moments on binary images using image block representation. Their idea was to use rectangular areas parallel to the image axis to represent the whole image and achieve faster computation of the moments. Flusser proposed a refinement of their method (Flusser (2000b)). In addition to using special data structures for blocks of the image, Flusser and Suk (1999) used precalculated repetitive tasks to speedup the calculations for video sequences. Although these methods were efficient for binary images, they are not easily extended for grey-scale images. The algorithms that need to follow the boundary of the objects are not suitable for the multiresolution analysis described in section 2.2.5.

Terrillon et al. (1998) used moment invariants to detect faces. They used skin colour segmentation to get blobs that were then split into sub-windows and their moments computed using the standard method. They used eleven moments, the original seven Hu's

set, plus four moments of 4th order, and used Neural Networks to train a simple classifier. There was no detailed analysis of the performance or accuracy. They extended their work to use other invariants (e.g. Fourier-Mellin moments in Terrillon et al. (2000)) as well as using the SVM (support vector machine) algorithm for training (Terrillon et al., 2004). Their work was limited to colour images, as the method was dependent on the segmentation of blobs to be able to work in real-time.

Only two references regarding the use of SATs and geometric moments were found. Schweitzer et al. (2002) used SATs with geometric moments and applied it to template matching. They used central moments (not Hu's set or any of the invariants) up to the 3rd order and used a match measure to compare different sub-windows to a single pattern (given by an image template). The other reference was the work carried out by Chen et al. (2005). They also used SATs up to the 3rd order to compute moment invariants. They applied their method to track the tip of a finger, achieving about 10000 moment computations per second. There was no attempt to train complex classifiers based on a set of images in any of the previous work.

The accuracy of the computation when detecting objects that rotate, scale or have different lighting conditions is another issue with feature extraction based on moment invariants. It has been known that noise affects high order moments (Teh and Chin, 1988), but even the accuracy of low order moments is also influenced by the transformations of the image itself. A good review on accuracy problems for various moment invariants is found in Rodtook and Makhanov (2005).

7.2 Geometric Moment Invariants

The proposed method uses several SATs to compute moment invariants. After extracting the moments, the values are trained with AdaBoost to produce cascade classifiers. Figure 7.1 shows an overview of the method.

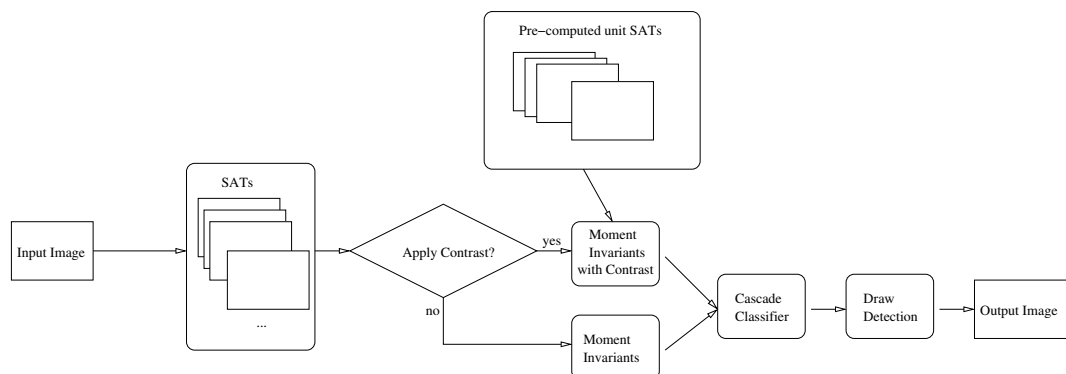


Figure 7.1: Method overview.

In this section, the equations for computing Hu's moment invariants using SATs are deduced. The approach allows rapid computation of moment invariants for any position and scale within an image. For clarity, the equations are presented in the form that they are usually presented in text books for image processing such as Gonzalez and Woods (2002) or Jain (1989). However, the focus of this section is the set proposed by Flusser (2000a). Higher order moments are known to be very sensitive to noise (Teh and Chin, 1988), and for that reason this work is limited to extract moments up to the 4th order. There are only eleven independent moment invariants up to the 4th order (Flusser, 2000a).

7.2.1 Hu's equations

Given a digital image $i(x, y)$, the 2D moment of order $(p + q)$ is:

$$m_{pq} = \sum_x \sum_y x^p y^q i(x, y) \quad (7.1)$$

For any order $(p+q)$, each element can be precomputed by multiplying the pixel value by its position. It is trivial to create SATs for 2D moments of any order. 2D moments are non-invariant, but they are the basis for Hu's equations. The values \bar{x} and \bar{y} are given by:

$$\bar{x} = \frac{m_{10}}{m_{00}} \quad (7.2)$$

and

$$\bar{y} = \frac{m_{01}}{m_{00}} \quad (7.3)$$

The central moment μ_{pq} is defined by:

$$\mu_{pq} = \sum_x \sum_y (x - \bar{x})^p (y - \bar{y})^q i(x, y) \quad (7.4)$$

And the normalised central moment η_{pq} is given by:

$$\eta_{pq} = \frac{\mu_{pq}}{\mu_{00}^\gamma} \quad (7.5)$$

Where $\gamma = \frac{p+q+2}{2}$

The following equations are for the seven 2-D moment invariants proposed by Hu,

from now on referred to as ϕ_n . They are invariant to translation, scaling, rotation, and mirroring:

$$\phi_1 = \eta_{20} + \eta_{02} \quad (7.6)$$

$$\phi_2 = (\eta_{20} - \eta_{02})^2 + 4\eta_{11}^2 \quad (7.7)$$

$$\phi_3 = (\eta_{30} - 3\eta_{12})^2 + (\eta_{03} - 3\eta_{21})^2 \quad (7.8)$$

$$\phi_4 = (\eta_{30} + \eta_{12})^2 + (\eta_{03} + \eta_{21})^2 \quad (7.9)$$

$$\begin{aligned} \phi_5 = (\eta_{30} - 3\eta_{12})(\eta_{30} + \eta_{12})[(\eta_{30} + \eta_{12})^2 - 3((\eta_{21} + \eta_{03})^2)] \\ + (3\eta_{21} - \eta_{03})(\eta_{21} + \eta_{03})[3(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2] \end{aligned} \quad (7.10)$$

$$\phi_6 = (\eta_{20} - \eta_{02})[(\eta_{30} + \eta_{12})^2 - ((\eta_{21} + \eta_{03})^2)] + 4\eta_{11}(\eta_{30} + \eta_{12})(\eta_{21} + \eta_{03}) \quad (7.11)$$

$$\begin{aligned} \phi_7 = (3\eta_{21} - \eta_{03})(\eta_{30} + \eta_{12})[(\eta_{30} + \eta_{12})^2 - 3((\eta_{21} + \eta_{03})^2)] \\ + (3\eta_{12} - \eta_{30})(\eta_{21} + \eta_{03})[3(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2] \end{aligned} \quad (7.12)$$

Flusser (2000a) recommended that Hu's moments ϕ_2 and ϕ_3 should not be used in 2-D object recognition, as they are dependent on the others. An extra independent 3rd order moment is given by:

$$\phi_8 = \eta_{11}((\eta_{30} + \eta_{12})^2 - (\eta_{03} + \eta_{21})^2) - (\eta_{20} - \eta_{02})(\eta_{30} + \eta_{12})(\eta_{03} + \eta_{21}) \quad (7.13)$$

The invariant set proposed by Flusser (2000a) was adopted in this work, limited to moments of 4th order. The expression to compute them directly from the normalised central moments η_{pq} is deduced, so they can easily be implemented with SATs¹. In the complete set used in this work, five of the moments are part of the original Hu's set, a new 3rd order moment and five 4th order moments from Flusser (2000a) as follows:

$$\psi_1 = \eta_{20} + \eta_{02} \quad (7.14)$$

$$\psi_2 = (\eta_{30} + \eta_{12})^2 + (\eta_{03} + \eta_{21})^2 \quad (7.15)$$

$$\begin{aligned} \psi_3 = & (\eta_{30} - 3\eta_{12})(\eta_{30} + \eta_{12})[(\eta_{30} + \eta_{12})^2 - 3((\eta_{21} + \eta_{03})^2)] \\ & + (3\eta_{21} - \eta_{03})(\eta_{21} + \eta_{03})[3(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2] \end{aligned} \quad (7.16)$$

$$\psi_4 = (\eta_{20} - \eta_{02})[(\eta_{30} + \eta_{12})^2 - ((\eta_{21} + \eta_{03})^2)] + 4\eta_{11}(\eta_{30} + \eta_{12})(\eta_{21} + \eta_{03}) \quad (7.17)$$

$$\begin{aligned} \psi_5 = & (3\eta_{21} - \eta_{03})(\eta_{30} + \eta_{12})[(\eta_{30} + \eta_{12})^2 - 3((\eta_{21} + \eta_{03})^2)] \\ & + (3\eta_{12} - \eta_{30})(\eta_{21} + \eta_{03})[3(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2] \end{aligned} \quad (7.18)$$

$$\psi_6 = \eta_{11}((\eta_{30} + \eta_{12})^2 - (\eta_{03} + \eta_{21})^2) - (\eta_{20} - \eta_{02})(\eta_{30} + \eta_{12})(\eta_{03} + \eta_{21}) \quad (7.19)$$

$$\psi_7 = \eta_{40} + \eta_{04} + 2\eta_{22} \quad (7.20)$$

$$\psi_8 = (\eta_{40} - \eta_{04})[(\eta_{30} + \eta_{12})^2 - (\eta_{03} + \eta_{21})^2] + 4(\eta_{31} - \eta_{13})(\eta_{30} - \eta_{12})(\eta_{03} - \eta_{21}) \quad (7.21)$$

¹The complete derivation of the equations 7.14 to 7.24 are included in the Appendix

$$\psi_9 = 2(\eta_{31} + \eta_{13})[(\eta_{21} + \eta_{03})^2 - (\eta_{30} + \eta_{12})^2] + 2(\eta_{30} - \eta_{12})(\eta_{21} - \eta_{03})(\eta_{40} - \eta_{04}) \quad (7.22)$$

$$\begin{aligned} \psi_{10} = & (\eta_{40} - 6\eta_{22} + \eta_{04})\{[(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2]^2 - 4(\eta_{30} + \eta_{12})^2(\eta_{03} + \eta_{21})^2\} \\ & + 16(\eta_{31} - \eta_{13})(\eta_{30} + \eta_{12})(\eta_{03} + \eta_{21})[(\eta_{30} + \eta_{12})^2 - (\eta_{03} + \eta_{21})^2] \end{aligned} \quad (7.23)$$

$$\begin{aligned} \psi_{11} = & 4(\eta_{40} - 6\eta_{22} + \eta_{04})(\eta_{30} + \eta_{12})(\eta_{03} + \eta_{21})[(\eta_{30} + \eta_{12})^2 - (\eta_{03} + \eta_{21})^2] \\ & - 4(\eta_{31} - \eta_{13})\{[(\eta_{30} + \eta_{12})^2 - (\eta_{03} + \eta_{21})^2]^2 - 4(\eta_{30} + \eta_{12})^2(\eta_{03} + \eta_{21})^2\} \end{aligned} \quad (7.24)$$

These moments are invariant to translation, scaling, rotation and mirroring. They are from now on referred to as ψ_n .

7.2.2 Computing Moment Invariants from SATs

The similarity between equations 2.2 and 7.1 shows that one can compute 2-D geometric moments directly from the SATs. In order to create the SATs, equation 2.4 is modified:

$$m_{pq}(x, y) = m_{pq}(x - 1, y) + m_{pq}(x, y - 1) - m_{pq}(x - 1, y - 1) + i(x, y)x^p y^q \quad (7.25)$$

One can find generically the 2-D moment $m_{p,q}(x', y', s)$ of a sub-window at (x', y') with scaling factor s . Let us consider two identical sub-windows that are located in different places in the image. The sub-windows have different 2-D moments for orders $p \geq 0$ and $q \geq 0$. However, their values are equivalent to computing the moments based on the same sub-windows padded by pixels of value zero. As the moment invariants, ψ_n , are translation independent, their values are the same for both sub-windows.

Equations 7.14 to 7.24 depend only on the twelve normalised central moments η_{11} , η_{20} , η_{02} , η_{12} , η_{21} , η_{30} , η_{03} , η_{22} , η_{31} , η_{13} , η_{40} and η_{04} . These, on the other hand, depend on μ_{pq} , which can be computed using a number of 2-D moments m_{pq} and, therefore, using SATs directly. The zeroth order μ_{00} corresponds to the simplest SAT, i.e., the equivalent to the Integral Image used by (Viola and Jones, 2001a):

$$\mu_{00} = \sum_x \sum_y (\bar{x} - x)^0 (\bar{y} - y)^0 i(x, y) = \sum_x \sum_y i(x, y) = m_{00} \quad (7.26)$$

For order 1,1 μ_{11} can be derived as follows:

$$\begin{aligned}\mu_{11} &= \sum_x \sum_y (x - \bar{x})^1 (y - \bar{y})^1 i(x, y) \\ &= \sum_x \sum_y (xy - x\bar{y} - \bar{x}y + \bar{x}\bar{y}) i(x, y) \\ &= \sum_x \sum_y xy i(x, y) - \sum_x \sum_y x\bar{y} i(x, y) - \sum_x \sum_y \bar{x}y i(x, y) + \sum_x \sum_y \bar{x}\bar{y} i(x, y)\end{aligned}$$

Both \bar{x} and \bar{y} are constant for a sub-window. Also each of the four factors can be expressed as a function of the corresponding SAT:

$$\mu_{11} = m_{11} - \bar{y} m_{10} - \bar{x} m_{01} + \bar{x}\bar{y} m_{00} \quad (7.27)$$

The central moment of order 1,1 for any sub-window based on the SAT computed over the entire image can now be computed. Analogous derivation can be made for the other μ_{pq} , for which only the final equations (as a function of m_{pq} , \bar{x} and \bar{y}) are presented here²:

$$\mu_{20} = m_{20} - \bar{x} m_{10} \quad (7.28)$$

$$\mu_{02} = m_{02} - \bar{y} m_{01} \quad (7.29)$$

$$\mu_{30} = m_{30} - 3\bar{x} m_{20} + 2\bar{x}^2 m_{10} \quad (7.30)$$

$$\mu_{03} = m_{03} - 3\bar{y} m_{02} + 2\bar{y}^2 m_{01} \quad (7.31)$$

$$\mu_{12} = m_{12} - 2\bar{y}m_{11} - \bar{x}m_{02} + 2\bar{y}^2m_{10} \quad (7.32)$$

$$\mu_{21} = m_{21} - 2\bar{x}m_{11} - \bar{y}m_{20} + 2\bar{x}^2m_{01} \quad (7.33)$$

$$\begin{aligned}\mu_{22} &= m_{22} - 2\bar{y}m_{21} + \bar{y}^2m_{20} - 2\bar{x}m_{12} + 4\bar{x}\bar{y}m_{11} - \\ &\quad 2\bar{x}\bar{y}^2m_{10} + \bar{x}^2m_{02} - 2\bar{x}^2\bar{y}m_{01} + \bar{x}^2\bar{y}^2m_{00}\end{aligned} \quad (7.34)$$

$$\mu_{31} = m_{31} - \bar{y}m_{30} + 3\bar{x}\bar{y}(m_{20} - m_{21}) + 3\bar{x}^2(m_{11} - \bar{y}m_{10}) + \bar{x}^3(\bar{y}m_{00} - m_{01}) \quad (7.35)$$

²The complete derivation is found in Appendix C.

$$\mu_{13} = m_{13} - \bar{x}m_{03} + 3\bar{x}\bar{y}(m_{02} - m_{12}) + 3\bar{y}^2(m_{11} - \bar{x}m_{01}) + \bar{y}^3(\bar{x}m_{00} - m_{10}) \quad (7.36)$$

$$\mu_{40} = m_{40} - 4\bar{x}m_{30} + 6\bar{x}^2m_{20} - 4\bar{x}^3m_{10} + \bar{x}^4m_{00} \quad (7.37)$$

$$\mu_{04} = m_{04} - 4\bar{y}m_{03} + 6\bar{y}^2m_{02} - 4\bar{y}^3m_{01} + \bar{y}^4m_{00} \quad (7.38)$$

The central moments needed for the 11 independent moment invariants ϕ_n can be computed from the following 15 SATs: m_{00} , m_{10} , m_{01} , m_{11} , m_{20} , m_{02} , m_{12} , m_{21} , m_{30} , m_{03} , m_{04} , m_{40} , m_{22} , m_{31} , m_{13} .

Flusser (1998) proposed a method that considers that parts of the moment calculation are independent of the object. These parts can be pre-computed and the pixel values added to the calculation later. In a similar way, when the frame size is fixed, which is typical for video sequence processing, all the SAT calculations related to the positions of the pixels for the various orders can be computed only once at the initialisation. This can be especially useful for higher order moments.

7.2.3 Preliminary numerical experiments: rectangular areas

Experiment 1: performance computing moment invariants over rectangular areas

In this experiment, the first 6 independent moments (equations 7.14 to 7.19) were used to assess the potential of the performance for real-time systems. A grey-scale image was used to obtain the moments for sub-windows at different scales and positions. The time was measured to compare the direct method (computing the moments directly from the pixels of each sub-window) with the method using SATs. Table 7.1 shows results for a machine running Linux (kernel 2.4) with 512MB memory and a 2.4GHz processor. The image size was 176x144 pixels. The runtime values in table 7.1 reflect the mean of 10 measurements.

The speedups were of the order of 1000 times. It must be stressed that the speedup only occurs when using the method to compute more than one sub-window per frame. The method avoids the repetition of calculations over the same frame in order to obtain sums of values of a particular sub-window and results are encouraging in terms of performance. A second experiment was carried out to test the accuracy of simple cascade classifiers using moments, described in the next section.

Experiment 2: detecting a numerical hand-written character

This experiment has the objective of testing the detection of a simple pattern to assess the behaviour of the method in practice using a web camera.

A classifier was trained to find hand written 9s (and because the technique is rotation invariant, it should also find 6s). The classifier was trained using AdaBoost (algorithm

Table 7.1: Runtimes for moment invariants computation.

Translation factor	Scale factor	number of sub-windows	time for the normal method (sec)	time for the proposed method (sec)
2	1.1	175538	381.72	0.27
	1.2	161285	283.01	0.24
	1.3	148881	265.09	0.23
	1.4	138247	247.94	0.22
	1.5	129880	229.64	0.20
5	1.1	29035	40.98	0.05
	1.2	26640	39.85	0.04
	1.3	24628	38.90	0.03
	1.4	22900	37.19	0.03
	1.5	21372	34.01	0.03

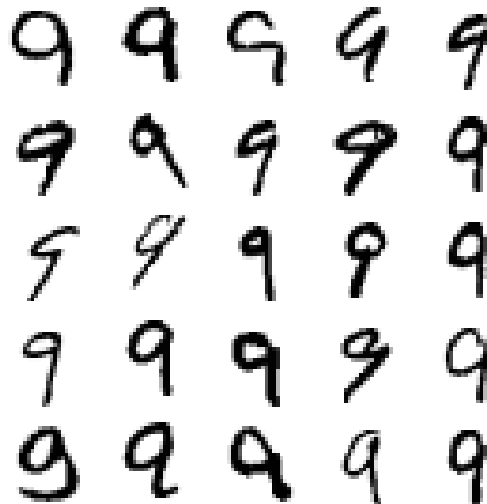


Figure 7.2: Grey scale samples of digit 9. The algorithm was trained with 9s, but it also detects 6s due to the rotation invariant characteristics.

3). The main difference is that, in our implementation, floating point numbers are used to store the feature values. A cascade was built using several rounds of AdaBoost, where new negative examples are presented for every new round. The final classifier was built using 500 positive examples (every round also had 500 negative examples) and 25 layers. The positive examples came from MNIST database (LeCun et al. (1998)). The characters in the examples occupied approximately 20x20 pixels on a 28x28 pixels square image (see figure 7.2). The negative examples were gathered randomly from the same images used in the previous chapters, making sure that no images presented the characters in the positive set. Both positive and negative sets are greyscale images.

Figure 7.3 shows a scene where several characters are randomly presented to a web camera. The rate varied from 1.2 to 3.0 frames per second (for a scaling factor of 1.5 and

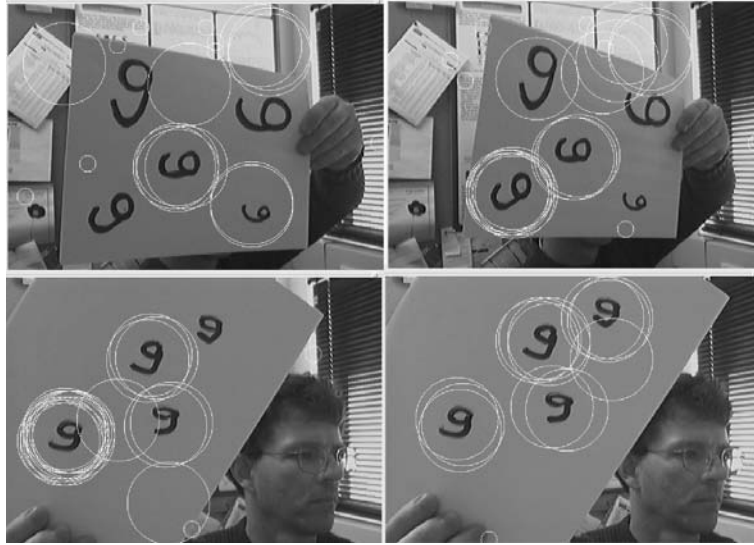


Figure 7.3: A sample of the working algorithm, showing the recognition of characters by images acquired by a simple web camera.

a translation factor of 5). The classifier used was not optimised to maximise positive hits and minimise false detections, but results in experiment 2 showed that it is possible to compute moment invariants in real-time using this method. Notice that there were several hits, some false and some regarding the same character in slightly different positions that should be merged.

7.2.4 Limitations of this method

During the training process, the feature values for the positive and negative sets are very close, indicating that the six 2^{nd} and 3^{rd} order moments might be insufficient to discriminate against random backgrounds.

In Viola and Jones (2001a), the features were not translation invariant and it was possible to train objects to be found in specific parts of the kernel. Using moments, however, it might happen that the object is located over a black background and that any of the sub-windows that contain the object is marked. A partial solution to this problem is to merge the positive hits that intersect each other.

Figure 7.4 shows a hypothetical situation that illustrates the problem. The three rectangular areas yield the same moments and, if within the margins of the classifiers, they are marked as hits.

7.2.5 Lighting Contrast Stretching

Contrast stretching is not applied to the entire image, but rather has to be applied to individual sub-windows. An update of all the SATs is required for every sub-window

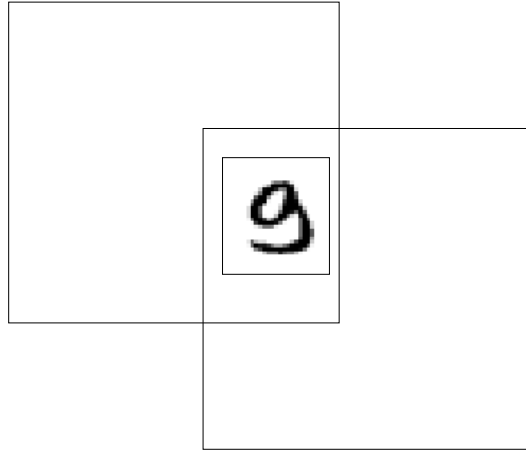


Figure 7.4: Merging positive hits with Moments Invariants over the same object demand a more sophisticate approach than the one adopted when using Haar-like features.

and the method becomes too slow. Ideally, the contrast stretching should be part of the feature extraction. An approximate method based on what Lienhart and Maydt (2002) did for Haar-like features is proposed. One needs extra SATs for a rapid implementation of the contrast stretching using moments due to the fact that there are different orders for the moments invariants. However, these extra SATs can be pre-computed as they are independent of the images. The basic equation used for the lighting contrast stretching in this chapter differs slightly from that used by Lienhart and Maydt (2002).

Lienhart and Maydt (2002) created an extra SAT using the squares of the pixel values ($I^2(x, y)$), so they could obtain the sum of the squares of any rectangular area of the frame. This SAT is used to compute the variance in rectangular areas of the image. With the mean and variance of a particular sub-windows computed straight from two SATs, they implemented a fast contrast stretching method. They used equation 2.8, adequate to the properties of the Haar-like features. Rather than using that equation, preliminary experiments showed that equation 7.39 is more convenient, as its form makes it easier to generalise the fast contrast stretching calculation:

$$\bar{i}(x, y) = \frac{255(i(x, y) - \mu + c\sigma)}{2c\sigma}, c \in \mathfrak{R}^+, \text{ and } 0 \leq \bar{I}(x, y) \leq 255 \quad (7.39)$$

Where $\bar{i}(x, y)$ is the image after contrast stretching, μ is the mean, σ is the variance, and c is a constant. This equation is similar to the one commonly used for contrast stretch (e.g. see Gonzalez and Woods (2002)). Rather than explicitly choose a value for the transform's constants (the slope and the crossing point of the straight line), these are based on the local mean and variance.

Each component $x^p y^q$ of a moment m_{pq} can be written as:

$$m_{pq} = \sum_n i(x, y) C_n \quad (7.40)$$

Where: C_n is $x^p y^q$.

The resulting moment after the contrast stretching is:

$$\bar{m}_{pq} = \sum_n \frac{255(i(x, y) + c\sigma - \mu)}{2c\sigma} C_n = \sum_n \frac{255i(x, y)C_n}{2c\sigma} + \sum_n \frac{255C_n(c\sigma - \mu)}{2c\sigma} \quad (7.41)$$

$$\bar{m}_{pq} = \frac{255}{2c\sigma} \sum_n i(x, y) C_n + \sum_n C_n (c\sigma - \mu) \quad (7.42)$$

$$\bar{m}_{pq} = \frac{255}{2c\sigma} (m_{pq} + (c\sigma - \mu) \sum_n C_n) \quad (7.43)$$

As C_n is a function of the position (x, y) only, the expression $\sum_n C_n$ can be pre-computed at the beginning using SATs with unit images (all pixels values set to 1) and do not need to be repeated for the duration of the sequence of images. Values between 1.5 to 2.0 are typical for the constant c . In our experiments, $c = 1.8$. The approach is not linear due to the cut off that needs to be done in order to limit the pixel values between 0 and 255. In practice, however, there are few pixels to be cut, so the final values for \bar{m}_{pq} given by equation 7.43 and 7.39 are equivalent.

Experiment 3: contrast stretching

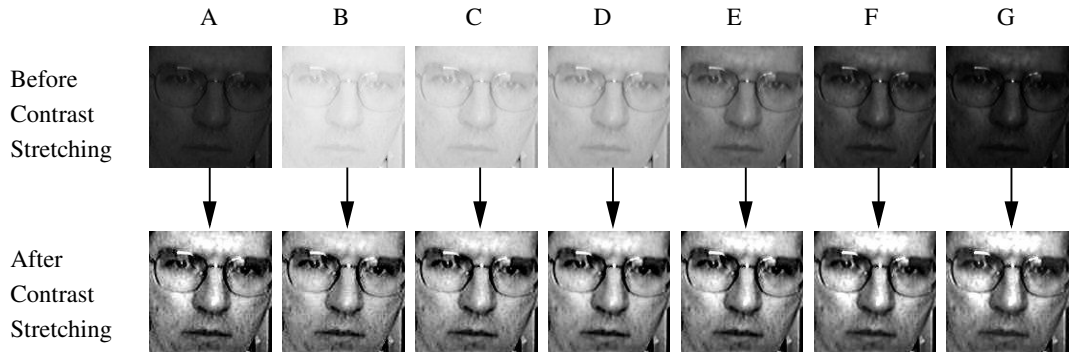


Figure 7.5: An example of contrast stretching. The images after applying the contrast stretching are actually slightly different to each other, yielding different moments.

The results of the computation of moments, with and without contrast, demonstrate that equation 7.43 works for a reasonably large variation in lighting conditions. Figure 7.5 shows some face images with different contrasts. The moments computed from these raw

images have large variances. Each image was pre-processed by equation 7.39, shown in the second row of images in figure 7.5. The moments computed from these stretched images are compared to the approximation given by equation 7.43 (computed directly from the raw images). Table 7.2 shows the mean and variance of the moment invariants obtained from the images of figure 7.5. The results show that the variances among the moments of the raw images are large, while the variances for both contrast stretching methods are much smaller. Moreover, the values for the moments computed from the stretched images from equation 7.39 are very similar to those computed directly from the raw images using equation 7.43.

Table 7.2: Variances for contrast stretching.

	raw images		equation 7.39 (slow method)		equation 7.43 (fast method)	
	μ	σ	μ	σ	μ	σ
ψ_1	6.56	0.7273	6.76	0.0048	6.73	0.0040
ψ_2	28.40	4.2471	26.28	0.3421	26.13	0.4193
ψ_3	57.02	7.0751	52.81	0.1389	52.33	0.1973
ψ_4	40.96	5.5388	39.43	1.7405	37.46	0.6608
ψ_5	56.03	8.5864	51.54	0.5968	51.28	0.7381
ψ_6	39.83	5.9100	36.61	0.4489	36.25	0.4529
ψ_7	12.82	1.4578	13.21	0.0118	13.15	0.0045
ψ_8	46.57	6.7813	43.91	0.9761	43.70	1.4094
ψ_9	44.98	6.5260	42.05	0.3995	41.66	0.3631
ψ_{10}	73.45	9.2288	70.60	1.5961	69.35	1.1996
ψ_{11}	73.13	10.4276	68.50	0.7519	68.14	0.8600

The results of this simple experiment showed that it is possible to overcome the problem of lighting variation when computing moments from SATs. The advantage of using equation 7.43 is the fact that the contrast stretching was incorporated into the moments computation, making it still possible to extract feature in real-time. The cost of the improvement is limited to additional lookups into the C_n SATs (fifteen SATs if computing the eleven moment invariants used before), which can be created at the start to be shared by any frame.

7.3 Concentric Discs

In this section, a new feature extraction method is proposed, using moment invariants (ψ_n) extracted from concentric discs of the area of interest. This method is called CDMI (Concentric Discs Moment Invariants). By computing moment invariants from concentric discs, 11 independent moments can be extracted from each different disc. This method increases the dimensionality of the training sets without losing its rotation invariance

properties.

The idea of extracting features from circular areas of the images is not new. To cite a few examples, Arof and Deravi (1998) used a circular neighbourhood to classify texture. Torres-Mendez et al. (2000) proposed a feature extraction method based on the number of intensity changes in pixels located at concentric discs. The centre of the circle was located on the centroid of the object, obtained via the moment of inertia of the image. Kazhdan et al. (2003) used concentric circles to compute symmetry descriptors for 2D images. Mukundan (2005) proposed the use of Radial Tchebichef Invariants (which are inherently computed over circular areas) for feature extraction and investigated their representation capabilities and their invariant properties. Another set of moments that are rotation invariant are the Zernike moments (for a comparative analysis see Chong et al. (2003)). However, the method proposed here is different from any of the work surveyed in the literature because it uses the special properties of the moments to achieve rapid feature extraction.

The number of independent moments up to the 4th order is limited to 11 (Flusser, 2000*a*). In order to improve the discrimination powers of the set, while keeping the rotation invariance property, a concentric circles approach is used in this section. This approach can improve upon the method described earlier on two fronts. Firstly, the dimension of the feature space is increased. Secondly, the scanning process does not suffer from the problem of translation invariance, making it easier to detect specific objects. This is especially important if the object is over a dark background, because dark patterns may easily make the classifier hit too many sub-windows. This can be achieved using the same pre-computed fifteen SATs used before, only requiring additional lookups. The resulting set of features is invariant to rotation and scaling, but not to translation.

7.3.1 Circular Area (Discs)

Normally, each rectangular sub-window requires only 4 table lookups per SAT (a total of 60 lookups if using the 15 SATs proposed here). In order to compute a circular sub-window, an approximation requires that small square areas are subtracted from the originally square sub-window (figure 7.6). The number of lookups can be minimised because there are common points among the smaller square areas. Some of the points are not at all necessary because they cancel each other out. The sum of the pixels in the area defined by the points 1,2,3 and 4, is given by:

$$A_{square} = pt_1 - pt_2 - pt_3 + pt_4 \quad (7.44)$$

If the 12 squares (a, b, \dots and l squares indicated in figure 7.6) are to be subtracted

from the large square that defines the sub-window, then A_{disc} :

$$\begin{aligned}
A_{disc} = & pt_1 - pt_2 - pt_3 + pt_4 \\
& - (pt_{a1} - pt_{a2} - pt_{a3} + pt_{a4}) - (pt_{b1} - pt_{b2} - pt_{b3} + pt_{b4}) \\
& \dots - (pt_{k1} - pt_{k2} - pt_{k3} + pt_{k4}) - (pt_{l1} - pt_{l2} - pt_{l3} + pt_{l4})
\end{aligned} \tag{7.45}$$

But there are common points among the square areas that cancel each other. Rewriting the equation 7.45:

$$\begin{aligned}
A_{disc} = & -pt_{a4} + pt_{b3} + pt_{c2} - pt_{d1} + pt_{e2} + pt_{e3} - pt_{e4} \\
& - pt_{f1} + pt_{f3} - pt_{f4} + pt_{g2} + pt_{g3} - pt_{g4} - pt_{h1} \\
& + pt_{h3} - pt_{h4} - pt_{i1} + pt_{i2} - pt_{i4} - pt_{j1} + pt_{j2} \\
& + pt_{j3} - pt_{k1} + pt_{k2} - pt_{k4} - pt_{l1} + pt_{l2} + pt_{l3}
\end{aligned} \tag{7.46}$$

And therefore, it suffices that 28 points are defined to compute the sum of pixels using all the 12 square areas (from a to l) in figure 7.6. Considering that eleven moments are computed using fifteen SATs, the total number of lookups per sub-window for the eleven moments is 420.

A classifier produced using Viola and Jones (2004) method for face recognition implemented in OpenCV (Bradski, 2000) had a total of 2913 Haar-like features distributed in 24 layers (cascades), requiring 17478 lookups. However, not all sub-windows reach the last layer, being eliminated by the classifier at an earlier stage. If a sub-window reached the 9th layer it would have used around 500 lookups, comparable to the method proposed in this work.

7.3.2 Concentric Discs computation

Figure 7.7 shows an example where a total of 66 moments (CDMI) can be computed. Each disc produces its own pattern, as it contains different pixels of the image. The method has the potential to improve the discrimination powers of the feature set.

The complete set of CDMI is not translation invariant. If the centre of the concentric discs is moved, the values for the internal discs differ from the ones computed previously. This property can be used to locate the exact position of an object, in the case of detection algorithms that use a moving kernel. This approach solves the problem shown in figure 7.4 for square or rectangular kernels.

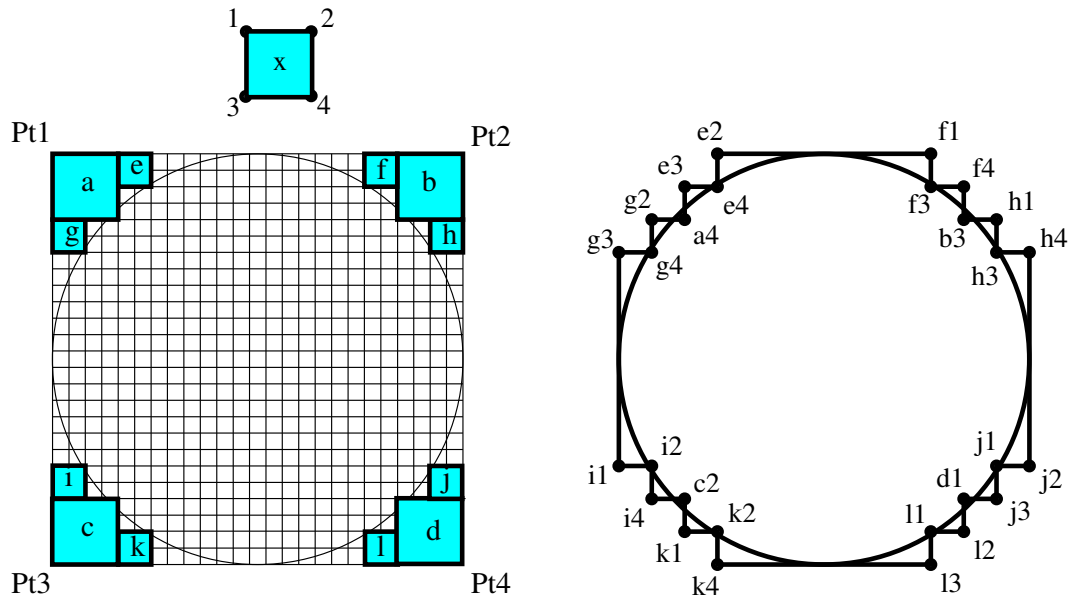


Figure 7.6: Computing the moments for an approximation of a circular area: the 12 square areas (in dark) are subtracted from each sub-window.



Figure 7.7: The concentric discs approach (CDMI): more features are extracted, as the areas within the inner circles get different patterns, enriching the feature set.

Experiment 4: accuracy under scaling and rotation

This experiment was carried out to verify the rotation and scaling invariance of the features extracted by this method. Firstly, the image in figure 7.8 (which can be found in Gonzalez and Woods (2002)) was used to compute the first 5 moments of Hu's set and the extra 6 moments from Flusser's set. The first 5 values might differ slightly from Gonzalez and Woods (2002) due to the precision of the variables used to create the integral images and used to compute the set. For comparison, the absolute values of the logarithm of the moments are computed. The results of simple scaling, rotation, and mirroring are shown in table 7.3.



Figure 7.8: Test image from Gonzalez and Woods (2002).

The same 5 images were used to compute the moments using the concentric discs approach. The discs' diameters are computed as a function of the width of the images (e.g., here 0.9 means that a circle with a diameter of 90% of the width was used). Table 7.4 shows the results for the disc with a diameter of 50% of the width. Table 7.5 shows the variance of the results for discs of various diameters (as per figure 7.7).

The results show that the approximation of the circular area works successfully, as the variance is small for most moments. The set of features can be used for recognition tasks where the rotation invariance property is important. Scaling invariance is maintained because the approximation of the disc is equivalent at different scales.

Figure 7.9 shows the actual approximations used to compute the discs for 50% diameter. The extra areas outside the circle create some variation on the moments values. A better approximation can be easily implemented by subtracting more areas. However, any improvement comes with the extra cost of more lookups in the SATs.

Table 7.3: 11 moments computed over a square area for image in figure 7.8.

	images					
	orig.	half	mir.	2°	45°	σ
ψ_1	6.600	6.600	6.600	6.596	6.595	0.0000
ψ_2	23.888	23.888	23.888	23.866	23.868	0.0001
ψ_3	49.200	49.201	49.200	49.152	49.134	0.0010
ψ_4	32.102	32.102	32.102	32.073	32.074	0.0002
ψ_5	47.850	47.850	47.850	47.807	47.810	0.0005
ψ_6	34.765	34.766	34.765	34.739	34.718	0.0005
ψ_7	12.838	12.838	12.838	12.830	12.829	0.0000
ψ_8	38.158	38.158	38.158	38.124	38.126	0.0003
ψ_9	40.248	40.250	40.248	40.220	40.197	0.0006
ψ_{10}	61.701	61.701	61.701	61.649	61.649	0.0008
ψ_{11}	61.978	61.978	61.978	61.930	61.924	0.0007

As expected, the larger variances in table 7.5 are associated with the high order moments. To compare the variances obtained with the approximation of a circular area, discs were cut (these are only as accurate as the scale permits) from the original images and measured the variance (see table 7.6). That would be the result if several additional smaller square areas were being subtracted from the image in such a way that the same pixels were involved in the computation of the moments. In other words, table 7.6 reflects the best case scenario for this set of images in the case of the concentric discs features.

As an indication of performance, 66 moments were computed from sub-windows of the image in figure 7.8. Using one processor of a dualcore AMD 2GHz, close to 20000 complete moment sets per second were computed (66 moments per sub-window at various scales and positions). This is much slower than the numbers shown in table 7.1, due to the fact that 66 moments from circular areas were computed, opposed to 11 moments from square areas. Also, circular areas required 28 lookups per CDMI, rather than just 4 lookups per moment. However, the code is not optimised and there is certainly room for improvement. Likewise in Viola-Jones method, the feature extraction can be fast enough to allow real-time detection applications to use this method. Training should also be easier due to the reduced number of dimensions when compared to the more than 100,000 features in a Haar-like feature set.

Table 7.4: 11 CDMI using an approximation of 12 square areas (with a diameter of 50% of the width) for image in figure 7.8.

	images					
	orig.	half	mirr.	2°	45°	σ
ψ_1	6.615	6.615	6.615	6.611	6.613	0.0000
ψ_2	25.225	25.243	25.225	25.201	25.192	0.0004
ψ_3	50.889	50.948	50.889	50.857	51.702	0.1311
ψ_4	34.151	34.179	34.151	34.110	34.143	0.0006
ψ_5	50.352	50.390	50.352	50.293	50.454	0.0035
ψ_6	35.362	35.364	35.362	35.363	35.376	0.0001
ψ_7	12.943	12.942	12.943	12.934	12.938	0.0000
ψ_8	40.352	40.379	40.352	40.305	40.344	0.0007
ψ_9	40.910	40.909	40.910	40.918	40.945	0.0002
ψ_{10}	66.087	66.270	66.087	66.052	66.161	0.0077
ψ_{11}	67.417	67.407	67.417	67.811	67.425	0.0311

Table 7.5: Variances for the approximation of concentric disc features from 1 to 0.5 in diameter.

	diameter					
	1	0.9	0.8	0.7	0.6	0.5
σ_{ψ_1}	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
σ_{ψ_2}	0.0001	0.0001	0.0002	0.0001	0.0001	0.0004
σ_{ψ_3}	0.0010	0.0003	0.0047	0.0015	0.0112	0.1311
σ_{ψ_4}	0.0002	0.0004	0.0012	0.0007	0.0047	0.0006
σ_{ψ_5}	0.0005	0.0007	0.0016	0.0026	0.0089	0.0035
σ_{ψ_6}	0.0005	0.0000	0.0047	0.4099	0.2052	0.0001
σ_{ψ_7}	0.0000	0.0000	0.0001	0.0000	0.0000	0.0000
σ_{ψ_8}	0.0003	0.0007	0.0021	0.0010	0.0078	0.0007
σ_{ψ_9}	0.0006	0.0007	0.0346	0.2224	1.6947	0.0002
$\sigma_{\psi_{10}}$	0.0008	0.0034	0.0221	0.2331	0.1935	0.0077
$\sigma_{\psi_{11}}$	0.0007	0.0014	0.0176	1.1691	0.0192	0.0311

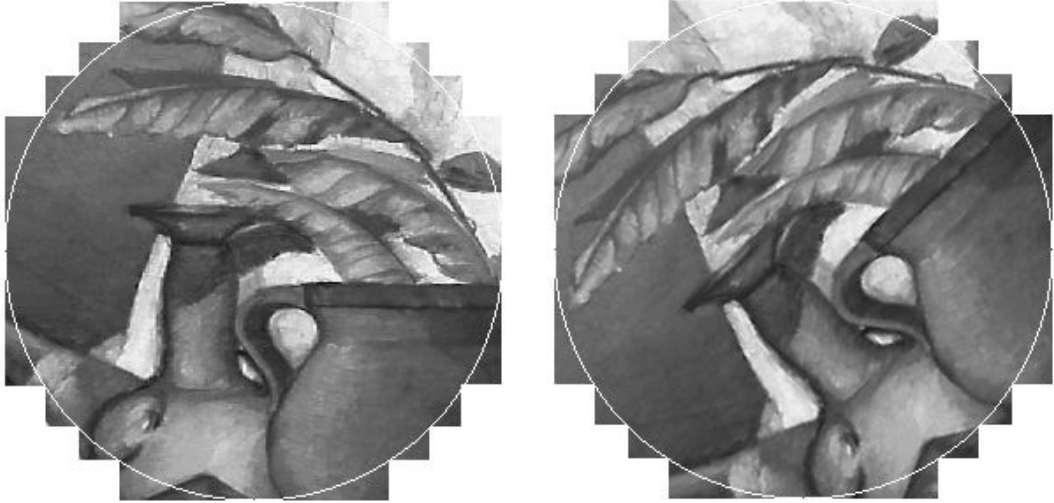


Figure 7.9: A 50% concentric disc area approximation for the original image and the 45° rotated image.

Table 7.6: Variances for the actual concentric disc areas (best case scenario, with the pixels cut directly from the images) from 1 to 0.5 in diameter.

	diameter					
	1	0.9	0.8	0.7	0.6	0.5
σ_{ψ_1}	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
σ_{ψ_2}	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001
σ_{ψ_3}	0.0010	0.0007	0.0008	0.0007	0.0011	0.0004
σ_{ψ_4}	0.0002	0.0002	0.0003	0.0003	0.0003	0.0003
σ_{ψ_5}	0.0005	0.0006	0.0006	0.0006	0.0006	0.0007
σ_{ψ_6}	0.0005	0.0001	0.0002	0.0002	0.0003	0.0002
σ_{ψ_7}	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
σ_{ψ_8}	0.0003	0.0003	0.0004	0.0004	0.0004	0.0003
σ_{ψ_9}	0.0006	0.0001	0.0001	0.0005	0.0005	0.0003
$\sigma_{\psi_{10}}$	0.0008	0.0007	0.0009	0.0010	0.0008	0.0010
$\sigma_{\psi_{11}}$	0.0007	0.0007	0.0007	0.0005	0.0004	0.0011

7.4 Face Detection

Using the methods developed in this chapter, classifiers were trained using AdaBoost for face detection. Two experiments were carried out. The first experiment used rectangular areas and 11 moments per sub-window. In this experiment, AdaBoost failed to produce good classifiers with low false detection, indicating that the set of features do not have enough discrimination powers to distinguish between the faces and backgrounds. The second experiment used concentric discs to compute 66 moments per sub-window. This experiment showed that the extension of the feature set is helpful, producing classifiers that can detect faces using a web camera.

7.4.1 Estimating angles using Hu's moments

Hu (1962) derived a method called The Method of Principal Axes where it is possible to estimate an angle based on some of the 2^{nd} order moments:

$$\theta = \frac{\arctan\left(\frac{2\mu_{11}}{\mu_{20}-\mu_{02}}\right)}{2} \quad (7.47)$$

There are restrictions that create a few problems for practical purposes. Firstly, $\mu_{11} > 0$ as well as $\mu_{02} \neq \mu_{20}$ for equation 7.47. If the object is completely free to appear at any angle, only an indication of 4 possible angles results. Depending on the shape of the object, the signals of μ_{02} , μ_{20} and μ_{11} may be used as an additional clues, but still would not tell angles that are 180 degrees apart. Secondly, the corner of the images plays an important role. When comparing two similar images taken in different angles, it is likely that the difference in the angle θ does not match the actual rotation. The problem can be appreciated in figure 7.10, where it shows faces extracted from the sequence Akiyo and rotated at various angles.

The resulting angle for the upright face is close to zero because this image has a strong symmetry along the y axis. For other objects there is a particular angle offset, depending on the distribution of the pixels. If the object has (as in the case of faces) a strong symmetry along one axis and a strong asymmetry along the other axis, it is possible to find the direction of the resulting axis and determine the angle. For example, the upright face yields an angle of 1.5° , while the upside down face yields an angle of 1.9° . The asymmetry (darker regions of the eyes) can be used to find the correct direction of the image.











	Nominal angle	Estimated angle
	0	1.5
	22	-20.6
	45	-43.5
	90	-1.6
	135	43.4
	180	1.9
	-135	-43.5
	-90	-1.9
	-45	43.4
	-22	23.5

Figure 7.10: Angles estimated for faces extracted from the first frame of the Akiyo sequence.

7.4.2 Classifiers for Face Detection using Moments

Experiment 5: face detection using 11 Moments

In this experiment, a classifier using 11 moment invariants (computed over square areas) was trained with AdaBoost. The positive images were taken from FERET and the negative images from the same source that was used in chapter 3. The training algorithm failed to get classifiers with low false detection rates, when using a large number of positive images.

The fact that the images are too ambiguous for classification indicates that the 11 moments, extracted from square sub-windows, are not enough to discriminate the positive and negative sets. The only reference to successful face detectors using moments relied on colour segmentation (Terrillon et al., 1998), making it easier to differentiate from the background. Even when using colour segmentation, Terrillon et al. (1998) found that there were a number of false detections that could not be overcome.

The experiment showed that the 11 moment invariants have a relatively poor discrimination characteristic for face detection applications.

Experiment 6: face detection using the CDMI approach

In order to show the potential of the method of concentric discs, real-time face detection was tested using two simple classifiers. A positive set with 250 face images acquired using a web camera was used. The negative set contained the same room's background to guarantee a low false detection rate. Figure 7.11 shows some samples of the application using a classifier produced with 250 positive examples.



Figure 7.11: Examples of successful detection with low false detection using 66 CDMI.

The face classifier produced with moment invariants is rotation invariant, as the examples in figure 7.11 shows. The performance is somewhat slower compared to the original Viola-Jones method. An average of 1 frame per second was achieved using resolutions of 480x640 pixels, with a kernel size of 128x128 pixels, a scaling factor of 1.1 and a translation factor of 3 pixels.

The experiment presented a number of interesting characteristics of the CDMI method. Firstly, the rotation invariance property holds well, as several rotated faces (at random) are correctly detected. Secondly, the scaling invariance property allows for some variation in the kernel size. Even though the original kernel size of the trained classifier is 128x128 pixels, detection with smaller kernels is achieved using the same classifier. In comparison, the Haar-like features only allow the use of kernels that are of equal or larger size than

the original size used during training. Thirdly, the training process is faster due to the limited dimension of the training sets.

Experiment 7: Measuring Accuracy with Small Negative Sets

The two previous experiments showed that the moment invariants do not have discrimination powers as strong as the Haar-like features. The attempt to train using the Viola-Jones version of AdaBoost did not yield good classifiers when using more than 500 faces samples from FERET.

In this final experiment, the number of negative sample images was limited to 20000 and trained the classifiers using algorithm 3 (chapter 3). The classifiers used in this experiment yielded too many false detections to be used in a real environment with random backgrounds, but they allow us to examine the issues regarding the accuracy for different dimensions of the training set.

The training process used up to 2000 face samples from FERET. The negative set was composed of 20000 images, randomly acquired from various images with no faces. Each classifier was tuned to keep the hit rates at 100% and trained up to 50 layers. Each layer was limited to 400 weak classifiers.

Figures 7.12 and 7.13 show the false detection ratio plotted against the number of weak classifiers used by the AdaBoost classifier. The figures show that training converges faster when using a larger number of CDMIs. Also, when more CDMI features are used in the training process, a smaller number of weak classifiers is needed in order to achieve the same level of false detections.

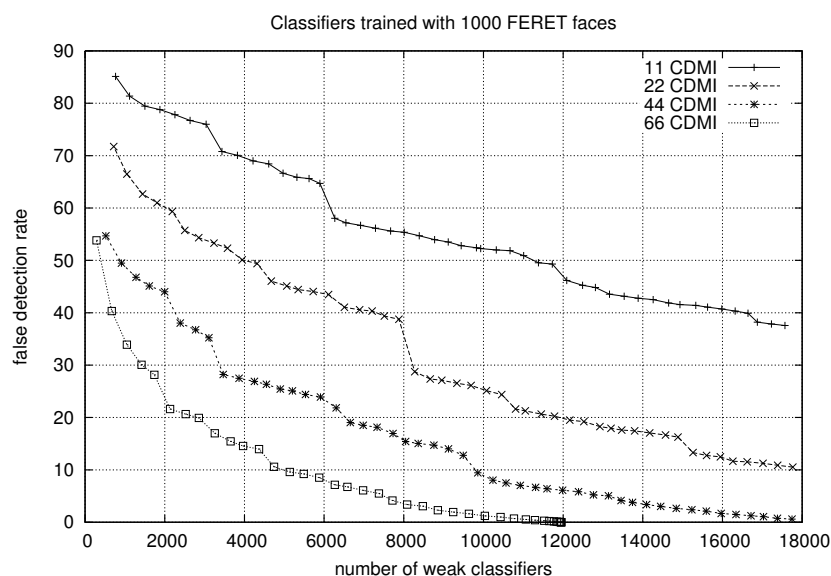


Figure 7.12: The false detection rate as a function of the number of weak classifiers. The positive set contains 1000 FERET faces and the negative set contains 20000 background images.

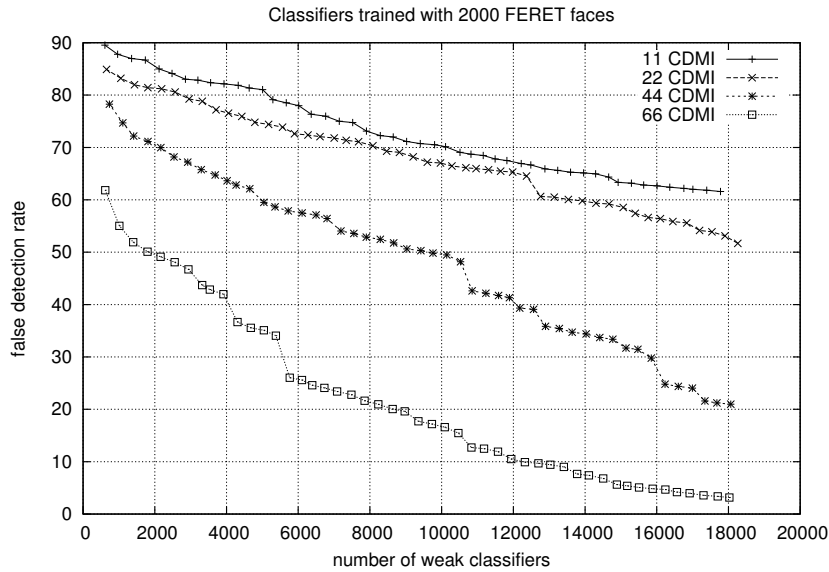


Figure 7.13: The false detection rate as a function of the number of weak classifiers. The positive set contains 2000 FERET faces and the negative set contains 20000 background images.

7.5 Summary

A new feature extraction method combining moment invariants (Hu, 1962; Flusser, 2000a) with SATs (Crow, 1984) has been presented. The method speeds up the computation of moment invariants over sub-windows acquired from a larger image and has the potential to be used in real-time computer vision algorithms. It is possible to implement the Viola-Jones method using Hu's features instead of Haar-like features. Besides the advantage of dealing with rotation invariant features, moment invariant features also limit the dimension of the feature set for the training sets.

Advantages of this method compared to Haar-like features:

- Rotation invariance (smooth detection when objects rotate in front of the camera).
- Faster training due to the limited feature space dimension.
- Flexible kernel size: due to the scaling invariance classifiers trained with a large kernel can be used with a proportional smaller kernel with a similar accuracy.

Some of the face classifiers trained for section 7.4.2 were accurate enough to be used in practice with a web camera, even though a generic face detection for the CMU-MIT dataset was not possible. The results are encouraging, but a question remained: is it possible to produce good classifiers for generic recognition of shapes? In chapter 8 the method is applied to a hand-written digit recognition problem, a very difficult problem due to the similarity among the images.

Chapter 8

Digits Recognition Using the Rapid Moment Extraction Method

This chapter presents the results of experiments with digit recognition using two different types of features. The first type used the normalised central moments (η_m), which are scale invariant but not rotation invariant. The second type used the method of the concentric discs (CDMI) developed in chapter 7 (based on Flusser's set of moment invariants ψ_n), which are both scale and rotation invariant. The experiments were constrained to handwritten digits, but the concepts seen here can also be applied to general OCR problems. The main question addressed in this chapter is whether the proposed set of features are discriminative enough to cope with handwritten characters. The scope of the study is limited to handwritten digits using a standard set of images collected by NIST.

Classifiers were trained using a modified version of AdaBoost. The classification problem presented in this chapter is of a different nature from the face detection problem. In face detection, the training is carried out considering a set of positive images against "the universe" of images. In practice, in each AdaBoost's round new negative samples are added, making the next stage training more difficult. The opposite is true in digit recognition, i.e., a number of negative samples are eliminated as new stages are added to the classifiers, until no negative samples are left.

The results showed that the discriminative powers of feature based on moments were not strong enough to create classifiers as reliable as the ones described in the literature. The best results achieved by these experiments were just below 10% test error (based on the MNIST database). Considering that the method is scale invariant, very fast and simple to implement, there is a potential use as a first stage in recognition problems.

This chapter is organised as follows. Firstly, a brief literature review shows the state-of-the-art in handwritten digit recognition and points to the difficulties faced by most

methods in terms of accuracy and performance. The next section describes the methods used to extract the features and to train the classifiers for the experiments in this chapter. Next, the results for training experiments are shown and an analysis of the accuracy of classifiers is discussed. In the final section, a detailed analysis of the errors for individual classifiers is presented.

8.1 Related Work

There is a number of handwritten digits databases such as MNIST, USPS, NIST and others. MNIST has been used recently as a benchmark for OCR methods. The MNIST database was based on the NIST SD-3 and SD-1 databases. The training set contains 60000 digits and the test set contains 10000 digits and is publicly available (LeCun et al., 1998). The digits were normalised to fit a 20x20 pixels image and were centred in a final 28x28 pixels image.

The task of recognising handwritten characters in real-time is a very difficult one. One of the critical steps involves feature extraction. Usually, one has to choose a compromise among certain characteristics such as invariance, discriminative powers, dimensionality, and computational complexity of the feature set. However, the classification process needs features that contains enough information about the class, and that is where moment invariants have problems.

Wong et al. (1995) proposed a new set of invariants based on Hu (1962) that achieved good correct recognition rates for a simple OCR problem using printed characters. Moments were limited to lower order due to numerical instabilities and to noise sensitivity. Trier et al. (1996) did a survey of feature extraction methods for OCR applications. They concluded that while the printed characters recognition problem is relatively simple, the handwritten character recognition needs more sophisticated methods and it is much more difficult to train accurate classifiers. They reminded that most successful OCR systems needed at least 10-15 features. However, a larger number of features was needed to achieve better accuracies with handwritten digits. Liao et al. (1997) used moment invariants to build a Chinese character recognition system, where he noticed that characters that were too similar had to be grouped together in order to make a strong system.

Baluja (1999) has studied the problem of recognising rotated digits. He used three different methods to cope with rotated digits. The first method used an exhaustive approach and was not accurate. The second used a two step approach where a de-rotation neural networks was trained to return an angle for unknown digits, followed by a single neural networks that classified the de-rotated images. The third approach used the same de-rotation approach, but individual classifiers were trained for each digit. The last approach was the most accurate, achieving 93% recognition, although the method itself does not provide scale invariance.

Table 8.1: Results reported on the MNIST database.

Method	Authors	Reported Test Error
linear classifier (1-layer NN)	LeCun et al. (1998)	12.0%
2-layer NN, 1000 hidden units	LeCun et al. (1998)	4.5%
Euclidean nearest neighbour	Simard et al. (1992)	3.5%
Haar-like features and AdaBoost	Casagrande (2005)	1.3%
3-Stage NN-NN-SVM	Gorgevik and Cakmakov (2004)	0.83%
LeNet4 with distortions	LeCun et al. (1998)	0.7%
BoostMap and BoostMap-C	Athistos et al. (2005)	0.58%
Combination of the methods	Keysers (2006)	0.35%

When analysing the errors made by various methods in the task of digits recognition Suen and Tan (2005) found that some of the characters were so ambiguous that hardly any of the available methods could correctly classify them. They presented a list of 127 handwritten digits from MNIST as being very difficult (which already represents an error of 1.27%). They divided the most common errors into three categories:

- Category 1: **geometric similarity** (such as 4s and 9s, 0s and 6s etc). The errors in this category are very difficult to overcome because there is usually an undefined boundary between such digits in any feature space.
- Category 2: **noisy images**. The errors in this category are due to degraded images. Common problems include writing habits, thick pens or pens that fail to write part of the digit.
- Category 3: **images easily recognisable by humans**. Usually errors in this category are due to the feature extraction process or due to the training process. If the feature set is not discriminative enough, further training is unlikely to improve the results.

There are a number of reports using the MNIST database with a variety of methods. It is beyond the scope of this work to discuss them all in detail. A good review of various methods (about 30 methods) was presented by Keysers (2006). LeCun et al. (1998) discusses implementations of convolutional neural networks method (including his ‘LeNet’ method). Examples of overall results are shown in table 8.1. The errors varied between 12.0% and 0.35%. The methods that presented very low errors added geometrically transformed samples to the training set.

Summing up, moment invariants were used before in OCR with limitations in accuracy. Moment invariants of higher order are sensitive to noise, limiting the dimensionality of moment based feature sets. There are many feature extraction methods that were

successfully used in OCR, but most of the features are not invariant to scale or rotation, with the exception of Haar-like features that are invariant to scale. Most of the feature extraction methods surveyed were developed specially for OCR applications and it would be difficult to apply to different recognition problems. It is useful to extend the feature set based on moments, as these features are invariant to scale and rotation, fast and easy to implement.

8.2 Feature Extraction

8.2.1 The Concentric Discs Method (CDMI)

The CDMI approach, detailed in section 7.3, can potentially help on the recognition of digits in texts. The method is rotation invariant, scale invariant and mirror invariant. The set as a whole is not translation invariant. The areas from which the moments ψ_n are extracted are approximated to a disc (figure 8.1). One would not expect this set to train very well when facing pairs of digits such as 6-9 or 2-5, although the experiments showed some surprising results.

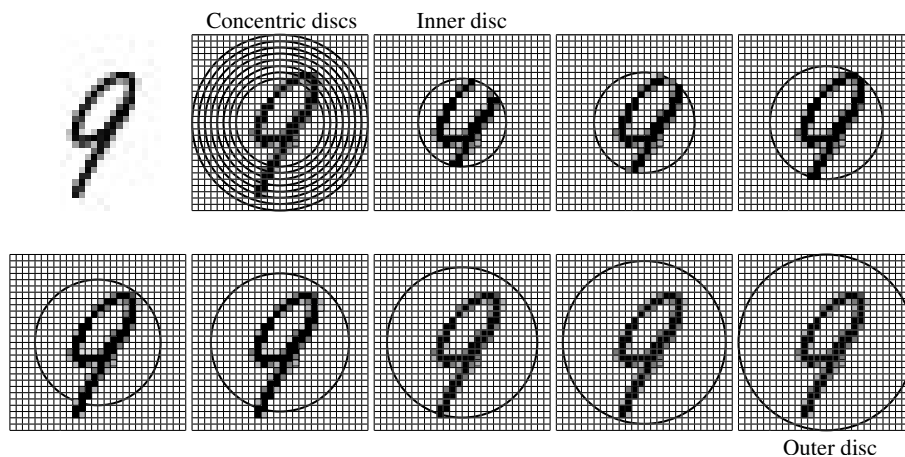


Figure 8.1: The CDMI approach: more features are extracted, as the inner discs may get different moments, enriching the feature set.

8.2.2 An Extended Feature Set for Normalised Central Moments

The normalised central moments η_{pq} (equation 7.5) are limited to 12 features per area, if using moments up to the 4th order. Computing moments from concentric square areas of the image produces a more complex set. In these experiments, a maximum of 7 square areas were used due to the limitation in the scale of the MNIST database, giving 84 independent features. This set is invariant to scaling, but not to translation (due to the concentricity constraints) or to rotation. Figure 8.2 shows the areas from where the

moments η_{pq} are extracted. All features are rapidly extracted based on the 15 SATs used before in chapter 7.

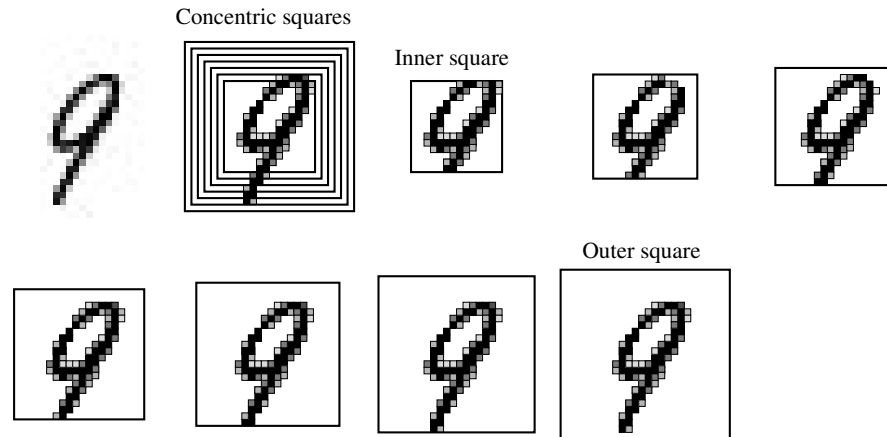


Figure 8.2: Computing 12 η s per square area.

8.3 Training Methods

Two different training approaches were used in this chapter, Convex Hull classifiers and AdaBoost. Early experiments using classifiers based on convex hulls (CH) showed the possibility to rapidly train using only positive examples (see section 3.1.2). If the feature space can form clusters out of the 10 digits, one can train a classifier almost instantaneously. The results of the CH experiments showed that, for handwritten digits, CH based classifiers are not accurate. They can, however, be used for pre-classification, leaving the more complicated job for other classifiers based on AdaBoost.

8.3.1 Convex Hull Classifiers

Convex hulls can be computed very fast for two dimensions (chapter 3). The training used a simple approach of combining all dimensions in such a way that it guarantees that all the positive examples are within the boundaries of all convex hulls. In practice, this means memorising only the vertices (a limited number of positive feature values) that composes the CH, which is much faster than AdaBoost.

During detection, an unknown digit belongs to the class if its features are contained by all CHs. In these experiments, a simple binary decision was used. It is also possible to use a more sophisticated approach considering the distance from the boundaries of the CHs. If the feature set is separable and there are enough samples, any negative sample eventually finds itself outside the boundary of one of the CHs and is eliminated in the process.

8.3.2 AdaBoost

A special version of AdaBoost was developed to achieve the highest possible hit rate and keep the false detection as small as possible. Due to the fact that both the positive and negative sets are limited, the AdaBoost version described in section 3.1.1, referred as algorithm 3, was used for these experiments. Figure 8.3 illustrates the modification. If a certain stage (of the cascade) does not achieve the specified hit and false detection rates, negative elements are discarded (and later re-added for further training). This process splits the negative set in more feasible portions, facilitating the training. In practice, two or more stages might intersect, as long as the resulting cascade classifier splits the samples correctly.

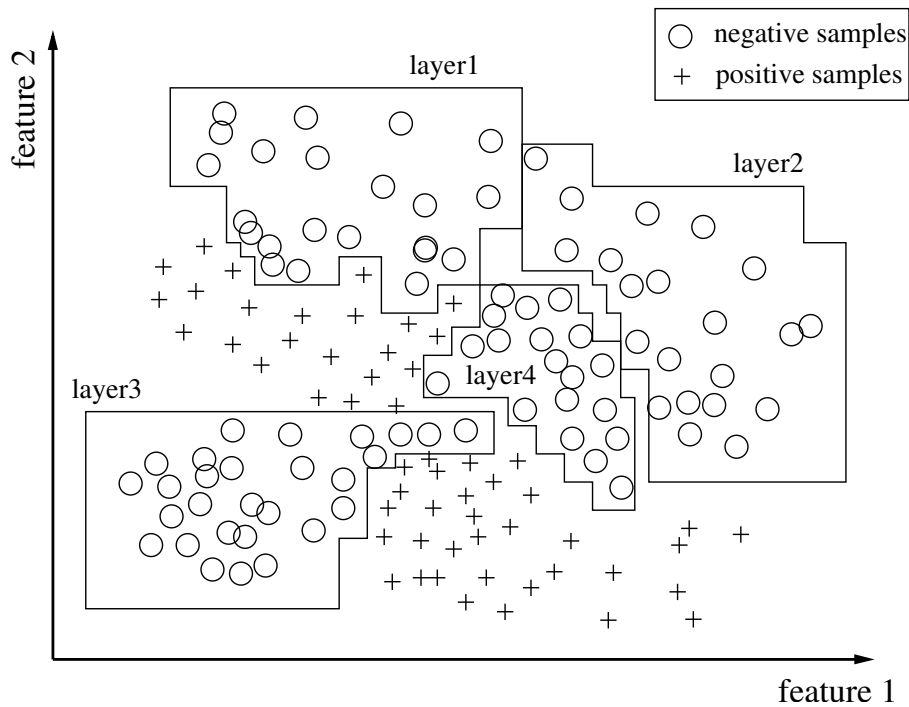


Figure 8.3: AdaBoost: working with sub-sets of the negative set, classifiers are specialised in certain areas of the feature space.

8.3.3 Confidence Value for Multiple Classifiers

All the classifiers trained with both methods are binary classifiers. An unknown digit can get a positive response by more than one classifier.

For the CHs, a simple Bayesian approach was used to choose between two or more positive results. The criteria used is the average of the distance of the point to the borders of the convex hull, multiplied by a confidence value found in the training error. Given the false detection $P(D|H_f)$ (probability that the classifier is positive, given that the digit is not d), using the classifier D_d on the training subset, the probability that the classifier is

correct is:

$$P(H_p|D) = \frac{P(H_p).P(D|H_p)}{P(H_p).P(D|H_p) + P(H_f).P(D|H_f)} \quad (8.1)$$

where $P(H_p|D)$ is the probability that the classification is correct for a certain digit d , if the classifier result is positive. The probability that the classifier results positive if the digit is in fact d is approximately 1, as the training process makes sure that no positives are missed ($P(D|H_p)=1$). The probability that a digit is d is $P(H_p)=0.1$ (one every 10 characters) and the probability that a digit is not D is $P(H_f)=0.9$ (nine of every 10 characters).

The likelihood for the CH can be associated with the distances from the point in the feature space to the borders of the CHs. If the average of the distances μ_{class} of a point to the borders of the CH is the same for two given classifiers, the classifier with the best confidence value wins.

$$Max(\mu_0.Co_0, \mu_1.Co_1, \dots, \mu_9.Co_9) \quad (8.2)$$

For the AdaBoost classifiers there is no measurable distance. Also, the confidence value for the case where all the classifiers reached zero training error is the same for all the binary classifiers. When faced with a situation where an unknown digit is positive for more than one classifier, the sum of the α coefficients for each weak classifier is used as a criteria to choose between positive results. When a certain image passes all the stages of a cascade classifier and yields a positive response, then the sum of the α s is bigger than every stage's threshold. These factors are linked to the confidence of the individual weak classifiers themselves (see Freund and Schapire (1999)). Considering that the larger the $\sum \alpha_n$ for a given stage, the better the confidence:

$$Max\left(\frac{\sum \alpha_0}{total(\alpha_0)}, \frac{\sum \alpha_1}{total(\alpha_1)}, \dots, \frac{\sum \alpha_9}{total(\alpha_9)}\right) \quad (8.3)$$

8.4 Experimental results

Three experiments were carried out for this chapter. The first experiment shows the difference in accuracy between four types of classifiers, using two different training methods and feature numbers, all for the feature type ψ_n . The second experiment used the best classifiers of the first experiment to carry out further training, until the training error was virtually zero. This determined the limitations of the accuracy of the classifiers produced by CDMI (feature type ψ_n). In experiment 3, feature type η_{pq} with AdaBoost was used. The training conditions used for feature type ψ_n in the two first experiments was repeated.

8.4.1 experiment 1: four types of classifiers

This experiment used four different types of classifiers, described next:

- **CH-11- ψ** : 11 moments per digit were computed and the set was used to train using CHs. The number of CHs are determined by the possible combinations of the 11 moments, 2 by 2 (therefore a total of 55 CHs).
- **CH-88- ψ** : 88 CDMI were computed and trained using CHs. Due to the large number of combinations, each set of 11 moments per disc is combined into 55 CHs independently, rendering a total of 605 CHs.
- **Ada-11- ψ -50**: 11 moments were computed and trained with AdaBoost. The AdaBoost was modified according to section 3.1.1 to guarantee that all positive examples are classified correctly.
- **Ada-88- ψ -50**: 88 CDMI were computed and trained with AdaBoost (with the same conditions Ada-11- ψ -50 training).

The training produced a single CH classifier per class (digit). Each classifier was tuned to find 100% of its class, with the penalty of high false detection rates. As the training only used positive samples, the training error measurement was carried out with each pair to compare to the AdaBoost classifiers.¹

The AdaBoost training produced one classifier for every pair, amounting to 90 classifiers for Ada-11- ψ -50 and 90 classifiers for Ada-88- ψ -50. Each classifier was trained up to a maximum of 50 stages (or less if the training error converged to zero before reaching this number of stages), each stage with a limit of 100 weak classifiers. Each classifier was tuned to find 100% of its trained class. The training error against each individual class is shown in table 8.2. Overall test errors for the AdaBoost classifiers are presented in table 8.4. False detection rates for the two groups of CH classifiers are reported in appendix C, tables C.1 and C.2. Overall confusion matrices for the two groups of AdaBoost classifiers are presented in appendix C, tables C.3 and C.4.

¹The training error was assessed against each class separately, so there were 9 results per class per classifier. In the case of CHs the same classifier was repeatedly used to assess the training error against each class

Table 8.2: Training errors for four feature sets.

Classifiers		classes									
		0	1	2	3	4	5	6	7	8	9
0	CH-11- ψ	-	20.2	66.0	71.5	56.8	68.7	15.0	9.9	91.9	35.6
	CH-88- ψ	-	2.4	52.0	57.2	44.2	55.1	12.6	7.9	78.5	30.4
	Ada-11- ψ	-	0.4	44.8	43.9	24.5	61.0	5.7	4.5	71.5	17.0
	Ada-88- ψ	-	0	26.4	6.4	4.7	23.6	0	0	11.5	0.9
1	CH-11- ψ	11.0	-	70.8	81.9	81.0	78.6	26.2	40.9	82.9	62.6
	CH-88- ψ	1.5	-	29.2	40.3	35.4	42.4	11.1	10.3	59.8	22.9
	Ada-11- ψ	6.4	-	30.6	47.8	29.6	64.4	7.3	13.9	54.0	27.6
	Ada-88- ψ	0	-	5.5	5.7	1.5	14.4	0	0	13.8	0
2	CH-11- ψ	43.4	71.1	-	97.8	96.9	96.5	75.0	90.5	97.3	89.7
	CH-88- ψ	33.5	34.1	-	82.9	86.9	84.6	68.0	78.8	82.6	75.9
	Ada-11- ψ s	58.5	18.7	-	90.5	88.2	90.6	62.1	76.5	82.0	72.4
	Ada-88- ψ	20.6	0.1	-	55.1	66.7	74.8	38.6	46.3	48.2	36.4
3	CH-11- ψ	40.0	77.5	94.6	-	90.6	95.7	70.1	85.0	95.5	87.1
	CH-88- ψ	17.3	41.8	77.2	-	75.2	81.8	57.4	58.0	82.9	67.4
	Ada-11- ψ	44.2	26.6	87.4	-	66.3	89.8	54.3	58.8	82.1	68.2
	Ada-88- ψ	2.6	0	63.9	-	48.7	71.5	29.7	24.1	62.5	29.2
4	CH-11- ψ	47.5	42.0	94.6	94.0	-	92.7	67.5	79.4	94.7	90.2
	CH-88- ψ c	20.5	16.6	63.8	64.1	-	67.3	54.4	51.3	82.7	78.9
	Ada-11- ψ	24.1	21.5	85.4	76.4	-	81.9	60.0	57.9	74.1	85.4
	Ada-88- ψ	1.3	0	63.9	38.5	-	62.8	29.2	34.0	37.8	51.2
5	CH-11- ψ	50.1	75.8	97.7	98.6	96.2	-	77.9	88.0	96.9	92.4
	CH-88- ψ	17.1	53.6	82.0	83.3	75.6	-	63.4	65.9	80.0	66.0
	Ada-11- ψ	61.5	41.0	88.0	90.1	77.3	-	52.7	64.9	87.0	70.6
	Ada-88- ψ	9.2	1.3	73.7	60.2	57.1	-	33.2	40.0	50.0	35.2
6	CH-11- ψ	31.8	8.6	83.6	91.0	76.0	77.8	-	84.5	65.8	94.5
	CH-88- ψ	16.9	3.9	63.2	66.7	62.6	59.8	-	68.2	53.5	76.7
	Ada-11- ψ	12.5	3.5	70.6	66.0	63.0	59.7	-	75.7	34.1	87.9
	Ada-88- ψ	0	0	39.0	24.9	38.5	30.6	-	52.9	12.1	49.8
7	CH-11- ψ	36.9	48.1	94.1	96.3	90.8	93.7	83.9	-	84.2	92.1
	CH-88- ψ	20.0	16.3	74.2	69.9	70.0	74.6	72.5	-	60.2	66.0
	Ada-11- ψ	21.3	14.0	81.9	76.3	75.3	79.9	79.6	-	51.6	87.4
	Ada-88- ψ	0	0	54.1	29.0	53.9	52.1	49.2	-	15.5	48.4
8	CH-11- ψ	39.4	70.8	91.6	93.2	93.4	89.5	57.9	55.9	-	82.6
	CH-88- ψ	15.0	39.0	47.4	56.2	74.0	56.4	36.5	23.4	-	64.6
	Ada-11- ψ	58.9	44.9	77.5	90.4	80.4	83.7	31.8	26.6	-	60.3
	Ada-88- ψ	3.9	1.1	35.1	48.9	58.0	55.4	12.9	8.1	-	23.1
9	CH-11- ψ	51.0	40.6	94.9	95.4	94.9	92.6	95.2	85.8	92.3	-
	CH-88- ψ c	29.6	13.9	65.5	67.2	38.8	67.0	83.4	59.6	81.4	-
	Ada-11- ψ	43.6	10.9	83.3	75.2	88.0	80.9	88.4	72.0	66.8	-
	Ada-88- ψ	12.6	0	40.2	33.9	68.8	49.8	58.5	33.3	39.4	-

Considering the high false detection rates, the CHs produced relatively inaccurate classifiers. Between the AdaBoost classifiers, clearly the one with higher dimensionality was easier to train. Some of the classes even yielded 0% training error before reaching the 50 stages. In the next experiment, the results of further training for Ada-88- ψ -50 are shown.

8.4.2 experiment 2: Ada-88- ψ with low training error

The Ada-88- ψ -50 classifiers were trained further until the training error converged to zero. This set of classifiers was called Ada-88- ψ -N. The confusion matrix is presented in table 8.3. The individual hit rates and false detection rates for each classifier are listed in appendix C, tables C.5 and C.6.

The hit rates (in italics) were below 50% for digits 2,3 and 5. The larger false detections (in bold) can, in many cases, be attributed to the fact that the feature set is invariant to mirroring and it is rotation invariant. For example, there is a large error associated with the pair 2-5. The error is smaller than it would be with printed characters because there are handwritten styles for digit 2 that presents at least two different patterns (a curled 2 opposed to a straight 2).

The overall results for the three sets of classifiers trained using AdaBoost (including

Table 8.3: Overall Confusion Matrix for Ada-88- ψ -N.

		predicted									
		0	1	2	3	4	5	6	7	8	9
Actual class	0	87.76	0	3.57	0.82	0.31	2.35	0.71	0.41	2.24	1.84
	1	0	92.6	0.18	2.2	0.09	1.5	0	0.35	3	0.09
	2	5.41	1.5	50.5	5.61	7.72	10.62	3.21	6.81	6.01	2.61
	3	0.59	0.69	10.00	59.5	1.68	8.51	3.47	2.48	10.89	2.18
	4	0.61	0.31	12.93	2.14	47.86	5.09	3.46	5.7	11.91	9.98
	5	3.36	1.23	18.83	8.63	3.14	39.35	2.47	10.09	8.3	4.6
	6	1.04	0.21	5.74	2.61	1.77	1.04	67.12	9.39	1.88	9.19
	7	0.39	1.17	6.32	2.14	3.02	3.11	9.44	67.7	0.58	6.13
	8	1.33	0.82	4.00	9.03	4.00	5.13	1.23	0.92	69.82	3.7
	9	1.88	0.30	2.18	1.88	4.96	2.87	9.32	3.67	2.48	70.47

Table 8.4: Test errors for feature type ψ_n (90 concurrent classifiers) for the 10000 digits MNIST test set.

Class	Ada-11- ψ -50	Ada-88- ψ -50	Ada-88- ψ -N
	Ada 11 feat 90 classif.	Ada 88 feat 90 classif.	Ada 88 feat. N stag 90 classif.
0	489	169	120
1	435	105	84
2	786	498	494
3	645	463	409
4	717	541	512
5	715	565	541
6	528	408	315
7	339	332	332
8	373	288	294
9	582	333	298
Total(%)	56.09	37.02	33.99

two from experiment 1) are presented in table 8.4. Comparing the three sets, there is a huge improvement in accuracy for the sets using a larger number of features. Training beyond the 50 stages improved the error only slightly, from 37% to 34%. The digits that achieved the best results were 0 and 1. The other digits did not get good accuracy, with digits 6,7,8 and 9 achieving average hit rates. The worst results were associated with digits 2,3,4 and 5.

For the best set of classifiers using type ψ features (Ada-88- ψ -N), a ROC curve for each class is presented (figure 8.4). The ROC curves were based on the joint results of the 9 classifiers per class. If all the 9 classifiers are hits, then that result is plotted as a hit. If any of the 9 classifiers for that class results in a false detection, then that result is plotted as a false detection. Because the negative sample sets are finite, the false detections are

plotted as percentages.

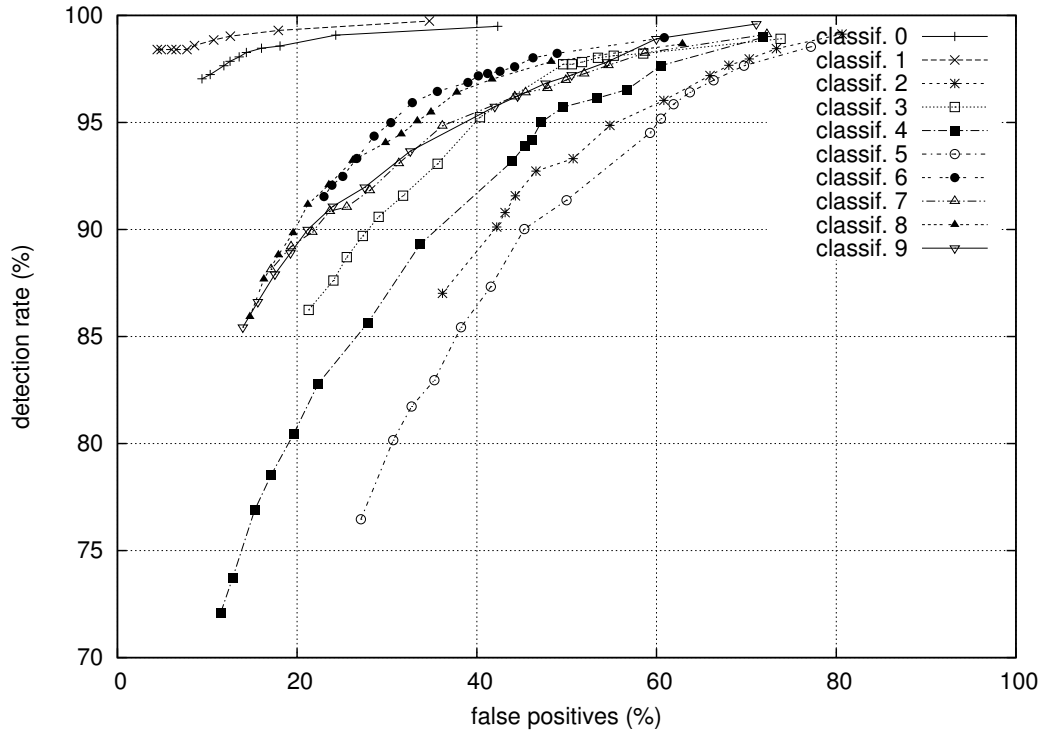


Figure 8.4: ROC curves for each classifier group, Ada-88- ψ -N.

8.4.3 Experiment 3: η features with AdaBoost

In this experiment, three sets of 90 binary classifiers were trained using η_{pq} feature type, with 12 and 84 features in the feature set. As per experiment 1 and 2, the first set was trained with 12 features and up to 50 stages (η -12-50). The second set was trained with 84 features, up to 50 stages (η -84-50). The third set also used 84 features but it was trained until the training error was virtually zero (η -84-N).

Table 8.5 shows the test results for η with 84 features N stages. The results collected for this table reflects the system as a whole, choosing the class with the best confidence value in the case of a draw between two or more classes. The classifiers worked well individually. The classifiers for the pair 0-1 achieved the best accuracy, with $class_{0-1}$ false detection at 0.4% and hit rate at 100.0%, and $class_{1-0}$ false detection at 0.0% and hit rate at 99.9%. The classifiers for the pair 4-9 achieved the poorest accuracy, with $class_{4-9}$ false detection 15.1% and hit rate 97.2%, and with $class_{9-4}$ false detection 13.0% and hit rate 97.0% (the individual results are presented in tables C.7 and C.8 in the appendix B).

The overall results for the three sets of classifiers are presented in table 8.6. There was a huge improvement in accuracy from the set of classifiers using 12 features to the one using 84 features, both using up to 50 stages. Further training the classifiers with

Table 8.5: Overall Confusion Matrix for η with 84 features (η -84-N).

		predicted									
		0	1	2	3	4	5	6	7	8	9
Actual class	0	<i>98.47</i>	0.00	0.00	0.10	0.00	0.31	0.71	0.10	0.31	0.00
	1	0.00	<i>98.94</i>	0.18	0.35	0.00	0.18	0.09	0.00	0.18	0.09
	2	1.55	1.45	<i>85.76</i>	2.52	1.74	0.48	2.52	1.94	1.74	0.29
	3	0.20	0.20	1.68	<i>90.30</i>	0.20	2.48	0.00	2.57	1.19	1.19
	4	0.31	0.00	0.81	0.31	<i>89.51</i>	0.61	2.24	1.93	1.02	3.26
	5	1.91	0.90	0.56	3.36	0.34	<i>86.88</i>	1.46	1.46	2.80	0.34
	6	1.15	0.31	0.84	0.10	0.10	0.73	<i>96.56</i>	0.00	0.21	0.00
	7	0.39	0.88	1.07	0.78	0.58	0.10	0.10	<i>94.55</i>	0.19	1.36
	8	1.23	1.03	0.41	3.29	3.70	4.62	1.03	1.64	<i>81.93</i>	1.13
	9	0.00	0.59	0.30	0.79	10.01	0.89	0.20	5.15	1.68	<i>80.38</i>

84 features improved the accuracy only slightly, from 10.3% to 9.5%. The best results were achieved with the digits 0, 1, 6 and 7 (all with hit rates larger than 90%). The most difficult digits were 8 and 9.

Table 8.6: Test errors for feature type η_{pq} (90 concurrent classifiers) for the 10000 digits MNIST test set.

Class	η -12-50	η -84-50	η -84-N
	50 stages 12 features	50 stages 84 features	N stages 84 features
0	114	15	15
1	45	12	12
2	572	144	147
3	258	101	98
4	421	101	103
5	623	119	117
6	35	33	33
7	89	57	56
8	553	179	176
9	424	269	198
Total(%)	31.3	10.3	9.5

For the best set of η classifiers a ROC curve for each class is presented (figure 8.5). The ROC curves were based on the joint results of the 9 classifiers per class. If all the 9 classifiers are hits, then that result is plotted as a hit. If any of the 9 classifiers for that class results in a false detection, then that result is plotted as a false detection. The false detections are plotted as percentages because the negative sample sets are finite.

An interesting observation is that, in many cases during the detection phase, there is a choice to be made between classifiers. About half of the errors were caused by choosing

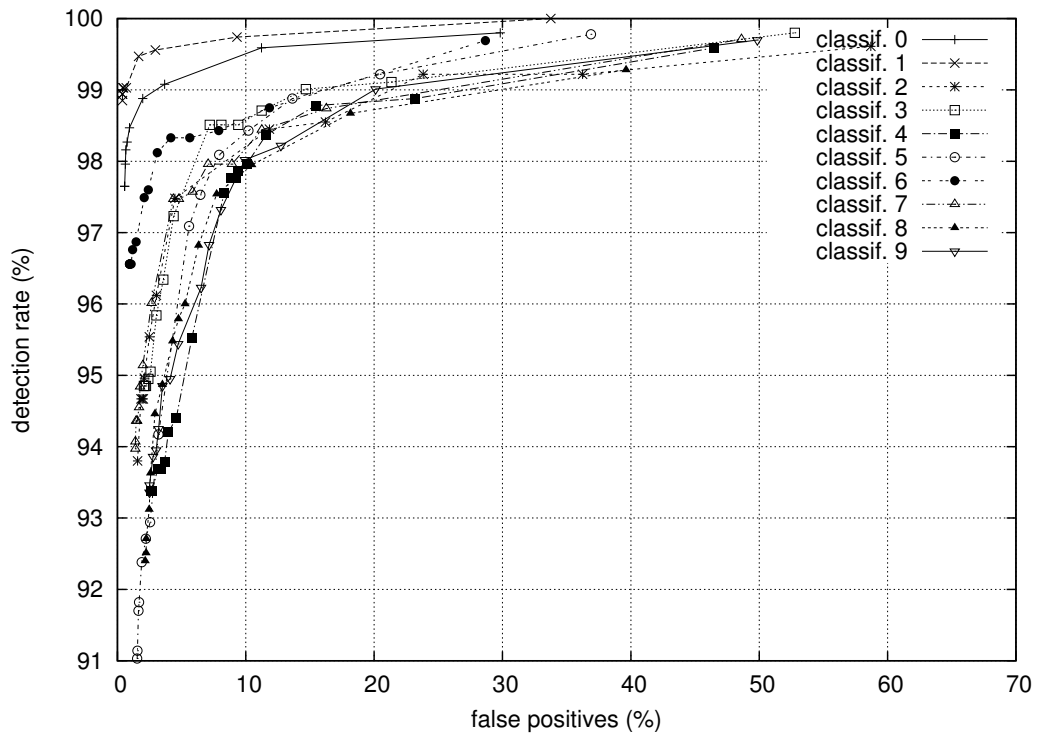


Figure 8.5: ROC curves for each classifier group, η -84-N.

the wrong digit out of two options available. Figure 8.6 shows the number of incorrect results per class when there was a draw between two or more classifiers. For example, the error caused by choosing the wrong class from a draw using η -84-N classifiers amounted to 5.46%, of the total error of 9.5%.

8.5 Discussion

For rotation/scale invariant features (ψ s), one would expect that the 6s and the 9s should be almost indistinguishable for this set of features, as the features are rotation invariant. The same for 2s and 5s, as the features are also mirroring invariant (a 2 may look like a 5 if it is upside down and mirrored). What the experiment demonstrates is that there are different patterns for these numbers for handwritten digits. One can hypothesise that the hand creates different patterns when writing these digits (for example the bottom strike of the 9s tend to be longer than the upper strike of the 6s).

For feature type η s the results were much better, mainly due to the fact that without the rotation invariance, these features present a better discrimination power than feature type ψ . The best result was just below 10% error rate. The results for the two types of feature are summarised in figure 8.7, where the accuracy per class can be compared.

One question raised during the experiments was: are there features that are being

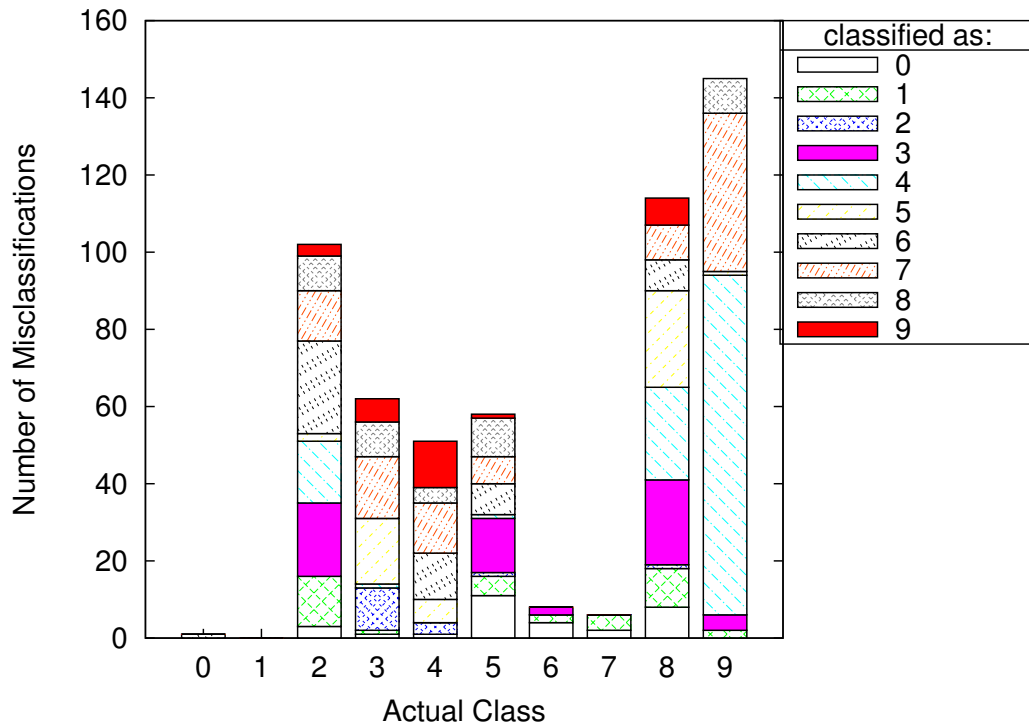


Figure 8.6: Number of errors caused by wrong choices between classifiers η -84-N.

chosen by AdaBoost more often than others? If a group of features were not being chosen at all, that would indicate that the set was over-representing the samples. The distribution of the usage of features for both η_{pq} and ψ_n is presented in figures 8.8 and 8.9. These figures were produced by counting the number of weak classifiers that use a certain feature in each of the 90 classifiers. The figures show that the features were all used by AdaBoost and were distributed, with few exceptions. For the η_{pq} features, feature number 7 (corresponding to η_{03} for the larger square area of the image) was the most used feature. For the ψ_n features, feature number 1 (corresponding to ψ_1 for the larger disc of the image) was the most used feature. There is also a visible cycle in the figures, indicating that for η_{pq} , features of orders 2 and 3 were used more often. For the feature type ψ_n , features of lower orders were used more often than higher orders.

One important aspect about the nature of images is illustrated in figure 8.10. Samples from the MNIST test set that can make a smooth transition between two digits were collected. For instance, a well handwritten 9 is very different than a 4. But when writing them quickly, one might fail to join the upper part of the digits 9 and straighten the bottom part in such a way that it looks like a digit 4. In practice, there is no defined boundary between some of the pairs of the handwritten digits, which explains part of the misclassifications. In the experiments, the sets were used rigorously according to the standard training approach, i.e., no additional training samples were created by distortion and

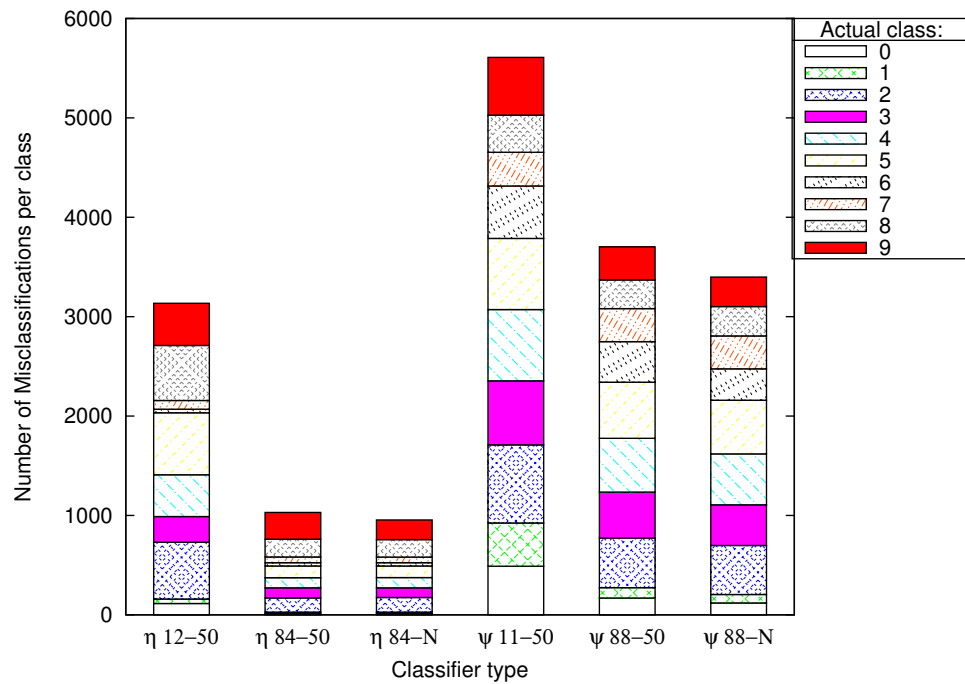


Figure 8.7: Compared misclassifications for the three η and ψ feature types.

no changes were made to the original MNIST images. Our training had a very low error rate (effectively 0%), and yet the test achieved poor results compared to the literature. However, no overfitting seems to have happened, as the test errors keep improving with the use of more stages in the classifiers.

A system based on moment invariants as described in this chapter is useful as a rapid first stage of handwritten digit recognition system. The first stage can identify the easier cases, and where there is a draw, a second stage of the system is put to work. Notice that, by limiting the number of stages in each cascade used at detection, most of the classifiers find the digit they are trained on, with the penalty of more false detections.

The experiments showed that there are intersections between competing classifiers that were not well defined due to a lack of samples. For example, the fact that classifiers 9-4 and 4-9² were both positive for an unknown character shows that there were areas of uncertainty that could not be trained (figure 8.11). With more samples, these areas can be minimised, but it is difficult to eliminate them completely. The intersection between classifiers indicates that the ambiguity between certain characters cannot be completely overcome with moment invariants.

Heuristics can play an important role in helping to eliminate misclassifications. In many cases, it was observed that, by adding another criteria to the set of moment invariants, it is possible to solve some of the ambiguities. It is possible to estimate the angle of

²Digit 9 trained with negative samples of digit 4 and vice-versa

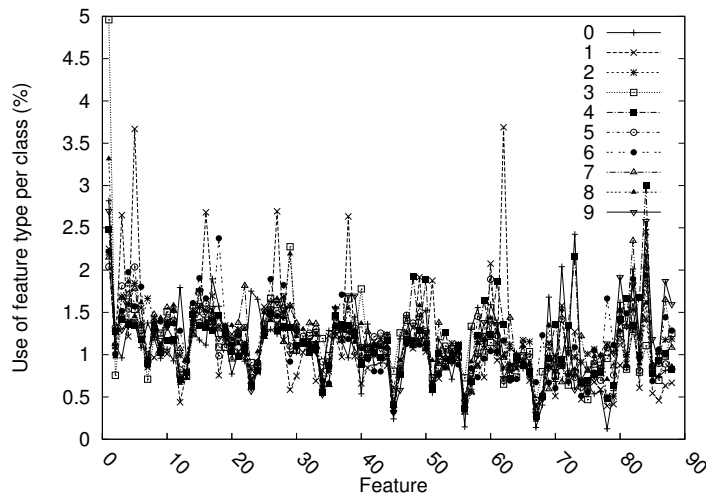


Figure 8.8: Distribution of the feature type usage per class for Ada-88- ψ -N.

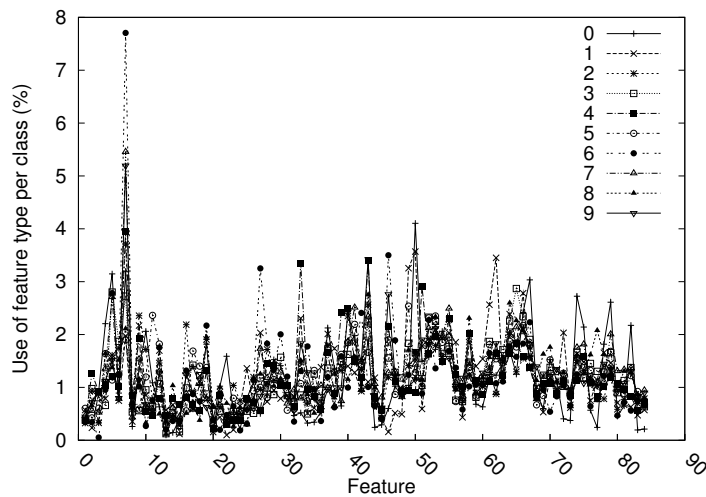


Figure 8.9: Distribution of the feature type usage per class for η -84-N.

the digit by using the approach described in Hu (1962). It is also possible to determine which part of the image contains more pixels. While keeping the rotation invariance, it is possible to align the digit and improve the chances of a correct classification.

Despite the performance in comparison with other methods' results, the moment invariants extraction method used here still has its merits. The η features are scaling invariant. The ψ s are scaling and rotation invariant. Both features can be computed from the same set of SATs, which makes it possible to run real-time applications. A combination of the two features can potentially make a stronger method.

		Classified as:									
		0	1	2	3	4	5	6	7	8	9
Label	0						0	0	0	0	
	1			1	1		1			1	1
	2	2	2		2	2	2	2	2	2	2
	3	3	3	3		3	3		3	3	3
	4	4		4			4	4	4	4	4
	5	5	5	5	5	5		5	5	5	
	6	6	6	6	6	6	6				
	7	7	7	7	7	7	7	7			7
	8	8	8	8	8	8	8	8	8	8	8
	9		9		9	9	9	9	9	9	

Figure 8.10: Examples of misclassifications for η with 84 features. Remarkably, some of the mistakes are the same obtained with LeNet-5 by LeCun et al. (1998), such as 0 classified as 7 and the 8 classified as 3.

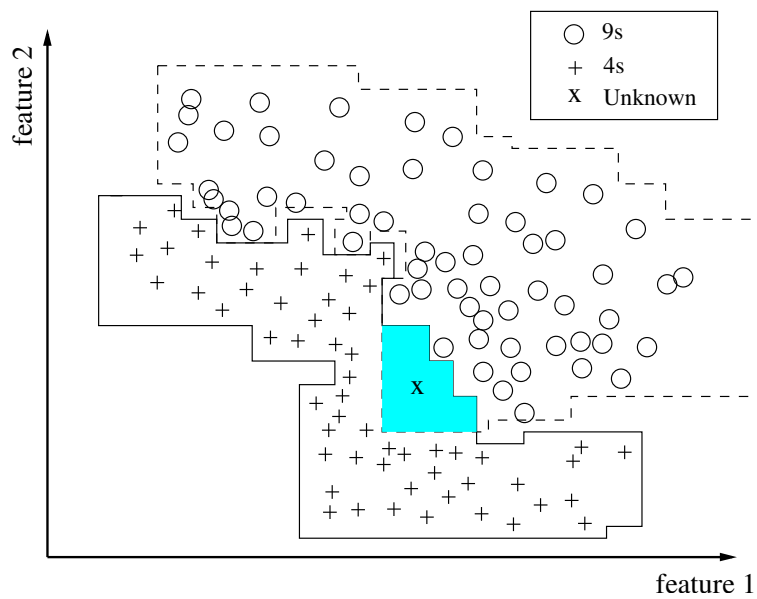


Figure 8.11: Intersection of two competing classifiers for an unknown character.

8.6 Summary

Two feature extraction methods for handwritten digits based on moment invariants were experimentally tested against MNIST. The η feature type (scale invariant but rotation non-invariant), achieved better results than the ψ type (scale and rotation invariant).

The experiments showed that, for handwritten digits, the CHs can only be used as a rough classifier.

The discrimination power of the moment invariants set up to the 4th order is not enough to train classifiers that can work reliably on their own, as all the methods tested are limited in accuracy. The best results, achieved with η feature types and 84 dimensions, was just below 10% error rate. Due to the computational characteristics and low training error, the methods can potentially be used as a first stage on a slower, more sophisticated and accurate application.

The main advantages of these feature extraction methods are speed and invariance to scaling. A promising future development is the combination of the two types of feature, as they are extracted from the same SATs.

Intriguingly, the training errors are very small for two groups of classifiers. There is no indication of overfitting, as the more training is carried out, the better the test error. Other training methods that do not yield such a small training error can be used to compare the accuracy with the AdaBoost algorithm. It would also be useful to train the same set of features with multiclass AdaBoost algorithms, recently discussed in the literature (Sun et al., 2006; Zhu et al., 2006).

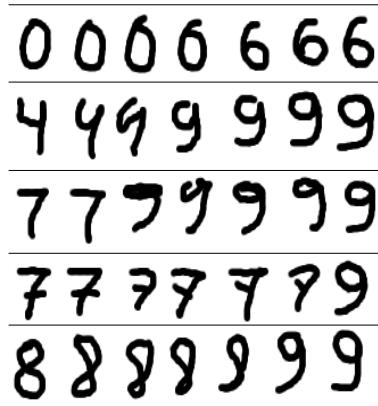


Figure 8.12: Hypothetical handwritten digits showing a gradual transition between certain digits.

Chapter 9

A Model for a Mobile Cluster Using Parallel Cascade Classifiers

This chapter describes the use of parallel machines to achieve better performance in object detection. Since the training algorithms were discussed before, this chapter limits the scope to the parallelisation of the detection phase using multiple cascades. Parallelisation can help to solve problems such as the need to run multiple separate classifiers. Multiple classifiers may be necessary, either because one wants to detect completely different objects simultaneously, or because an object needs different classifiers due to rotation or articulation.

Based on preliminary results, gathered from the deployment of two Beowulf clusters described in section 3.4, a mobile platform built with off-the-shelf components (hardware and software) is proposed. This platform can be fitted in a robot, car, backpack etc. A model of the performance of the platform using the Viola-Jones detector is presented. Simulation results are presented based on preliminary measurements using existing (static) clusters running Viola-Jones detectors.

In chapter 5, the efficiency of the Viola-Jones method using multiple classifiers for a simple gesture at various angles of rotation has been analysed. It has been observed that the use of concurrent cascade classifiers is very efficient because only sub-windows that are more likely to contain the detectable object are tested by all the cascades.

Standard parallel platforms are not necessarily suitable for real-time applications because of the limitations in bandwidth, the load balancing among the jobs, and their non-mobile characteristic. A mobile platform for object detection that can overcome these limitations is proposed. The platform should have the following characteristics:

- Cheap to build using commodity components.
- Easy to deploy and maintain.
- Low power consumption.

- Fast communication infra-structure among the processors.
- Allow a camera to be directly attached to the system.

This chapter discusses how these characteristics can be achieved and presents a model for the efficiency and performance of the proposed platform.

The relevant work discussed in the literature, as well as some preliminary tests with clusters, have been presented in section 2.3 and 3.3, respectively. This chapter is organised as follows. Firstly, a discussion about the proposed platform and its limitations is presented. The next section presents a model for the performance by making specific assumptions about the communication pattern. In the last section, this model is compared to empirical data obtained by running Viola-Jones detectors on existing clusters.

9.1 A Generic Platform for a Mobile Cluster

Since the first Beowulf cluster has been built (Sterling et al., 1995*b*), this approach has been the favourite among engineers and academics due to its flexibility, cost effectiveness, and performance. The platform proposed in this section follows this trend. However, good performance can only be achieved if the application's communication is minimised (Barbosa et al., 2001; Wilkinson and Allen, 1999). The hardware and software that compose the platform of the mobile cluster is briefly described next.

The hardware is composed of n processors, an Ethernet switch and a camera (figure 9.1). The processors are low power consumption boards with built in USB ports and Ethernet adapters. There are many alternatives to the traditional embedded boards that include dual fast Ethernet and even Gigabit Ethernet. Mini Ethernet Switches and USB cameras are nowadays standard devices, easily found in the market.

This cluster can be packed in a relatively small volume and powered by rechargeable batteries. Given the commercial alternatives available, a cluster with 8 nodes would occupy a volume of approximately 18000 cm^3 (or 18 litres), small enough to fit in a backpack or to be fitted into a car.

9.2 A Performance Model

9.2.1 Single Cascade

The performance of a single cascade depends on two parameters. The first parameter is the frame size, which affects mainly the SAT runtime. The second parameter is the total number of features computed by a set of cascades. These parameters depend on the cascade structure, as well as on the number of sub-windows being examined by the detector.

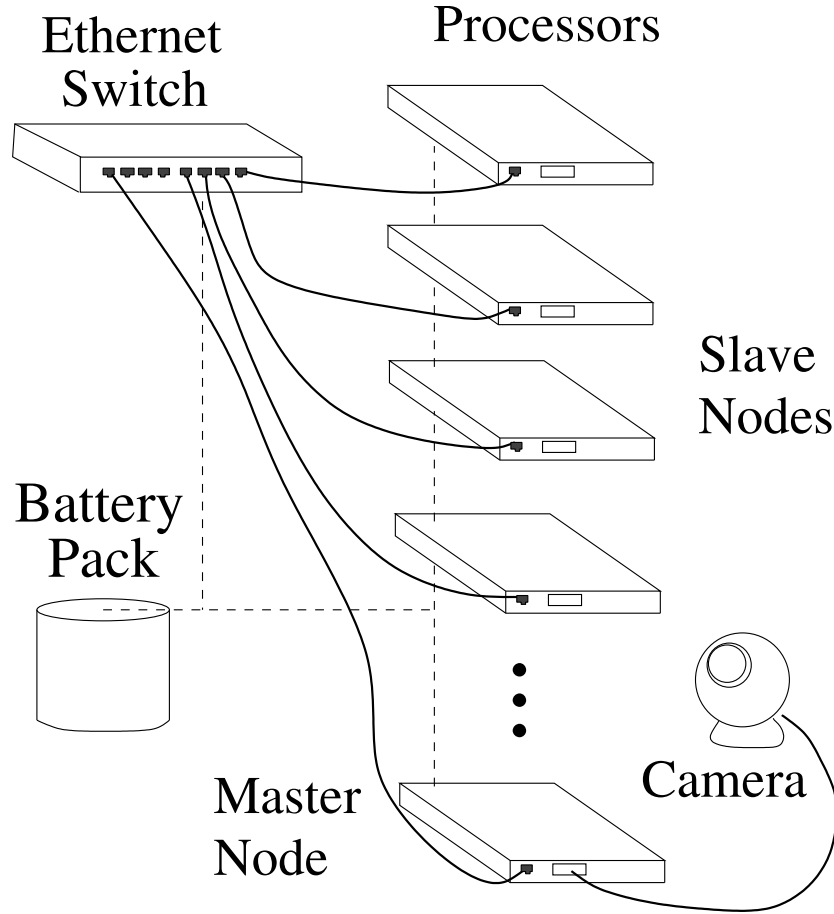


Figure 9.1: The proposed hardware platform.

The total number of sub-windows is a function of the frame size and two factors, the scaling factor and the translation factor. The following equation, combining equations 2.6 and 2.7, expresses the number of sub-windows to be examined by a single cascade:

$$Sub = \sum_{i=0}^n \frac{(W - M \cdot s^i)(H - N \cdot s^i)}{t} \quad (9.1)$$

where: W, H are the width and the height of the image, respectively. M, N are the width and the height of the kernel, respectively. t is the translation factor. s is the scaling factor. n is the maximum number of times the scaling is computed so that: $M \cdot f^n < W$ and $N \cdot f^n < H$

Each layer in a cascade eliminates a large portion of sub-windows that do not contain any object, so only a few sub-windows ever reach the last layer. The first few layers of the cascades only contain a few features. The last layers may contain several hundred features. To model the performance of each cascade, it is assumed that the growth of the number of features per layer is linear (the growth actually depends on the training constraints). Assuming that the number of sub-windows is fixed for an application (i.e.,

adopt an adequate value for the scaling and the translation factors), no more than $S(x)$ sub-windows have to be tested for each layer x of the cascade :

$$S(x) = 100 * (1 - F)^x \quad (9.2)$$

Where: F is the *false elimination rate*, a ratio of false sub-windows eliminated by layer x .

Actual figures can vary greatly depending on the image's nature, as well as on the training process. Figure 9.2 shows typical curves considering different average false elimination rates.

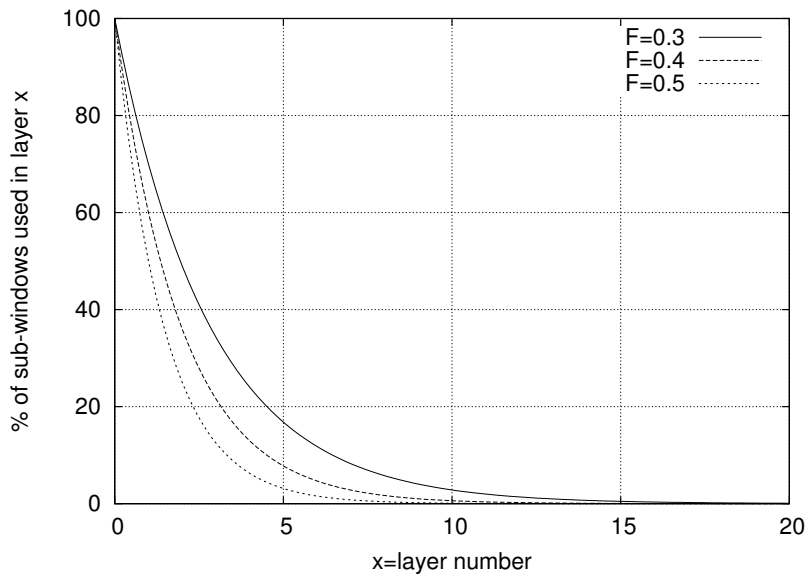


Figure 9.2: Percentage of sub-windows per layer.

Despite the fact that each layer has less sub-windows to test, the number of total feature computations can increase, as each sub-window needs a certain amount of features up to layer x . Many classifiers created by AdaBoost using Viola-Jones modified algorithm tend to increase the number of features per layer. The total number of feature computations at each layer is plotted in figure 9.3. The total number of features per layer is:

$$f(x) = Sub.(1 - F)^x.(f_0 + C.x) \quad (9.3)$$

Where: f_0 is the initial number of features in layer 0, Sub is the total number of sub-windows for a given frame (for a fixed translation and scaling factors) and C is a constant.

Finally, the sum of the features computed up to a certain layer is given by equation 9.4 and plotted in figure 9.4. Of the total number of feature computations at the end of a

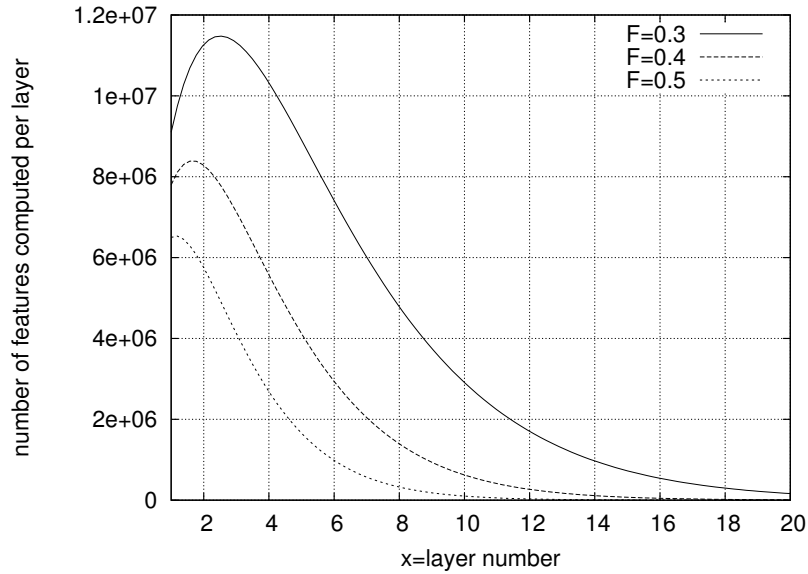


Figure 9.3: Features per layer for $Sub = 1000000$, $f_0 = 3$ and $C = 10$.

frame's detection, a large portion is done in the early stages. However, as seen in figure 9.2, the total number of computations decrease if the classifier is eliminating false sub-windows very aggressively.

$$S_t(x) = \sum_{l=1}^x Sub \cdot (1 - F)^l \cdot (f_0 + C \cdot l) \quad (9.4)$$

9.2.2 Concurrent Cascades

A simple geometric decomposition (or data decomposition as presented in section 2.3.4, see figure 2.8) for the parallel classifiers would not work well in this model. Once a SAT is computed, the classifier has to be used in a number of scales and positions. Some of these scales would have to overlap the geometric decomposition, so part of the computation would have to be reorganised. From this point on, it is assumed that the master node has to broadcast the whole frame (or broadcast a similar structure with information about the image) to the slaves. It is faster to send the frame rather than sending SATs, even if that means a repetition of the computation of SATs in the slaves, because the frame is smaller than its corresponding SAT. The frame can be converted to a 8 bit grey scale image. Each element in the SAT has to be larger than the elements in the frame to avoid overflow (greyscale frames uses 1 byte per pixel and SATs use 8 bytes per element).

A single processor is limited to run only a few classifiers with real-time constraints. It can be shown that Viola-Jones detector is suitable for parallelisation because it is possible to minimise the communication among the nodes. Cascades can share the same SATs when running concurrently over the same frame. Figure 9.5 shows that the detector only

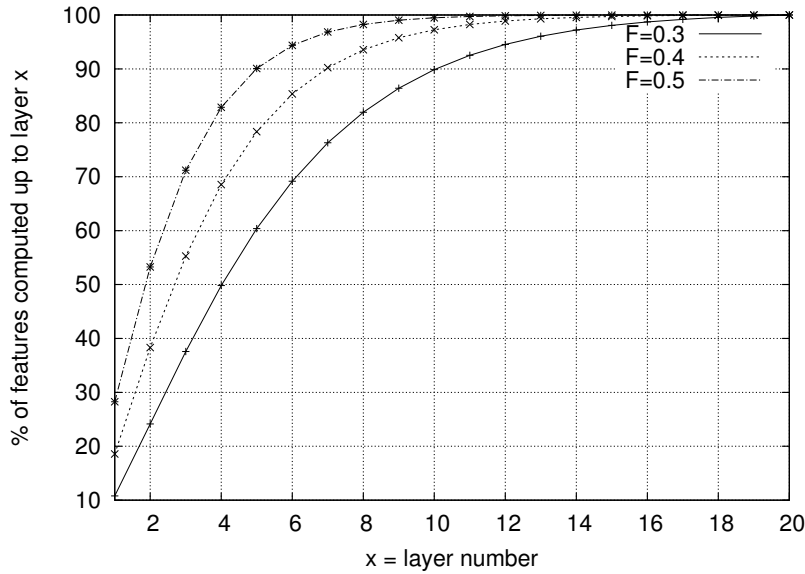


Figure 9.4: % of the features computed up to layer x .

examine the frame’s sub-windows after computing the SAT. The combined runtime (SAT + Cascades) determines the frame rates.

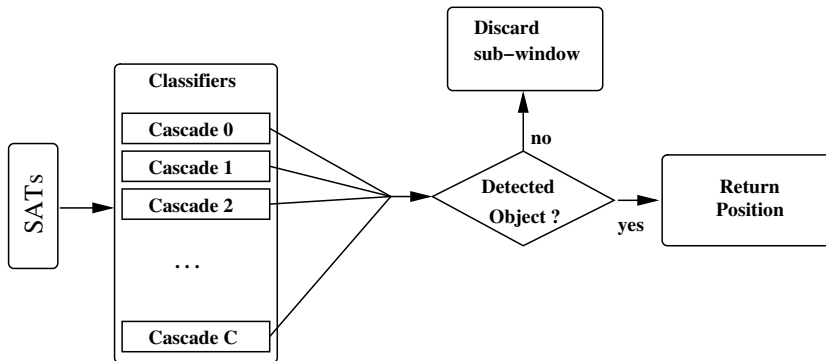


Figure 9.5: The detector using multiple cascades in one slave node.

Figure 9.6 shows three hypothetical runtime distributions, where the SAT runtime is constant and is smaller than the runtime of all cascades combined. The frame rate slows down logarithmically, as shown in figure 9.7. It must be stressed that figure 9.7 shows the performance of a single processor running concurrent cascades. Looking at the results of figure 9.7, one can determine what frame rate to expect, given the number of cascades and the runtime ratio. The figure also shows that for a reasonable frame rate, say between 5 to 10 frames per second, up to 5 cascades can run concurrently in the same node.

In order to show the effects of the computation of weak classifiers in a trained cascade, the results are plotted in figure 9.8. In this figure, a cascade trained for face detection was used in images with and without faces (5 frames chosen at random). The cascade

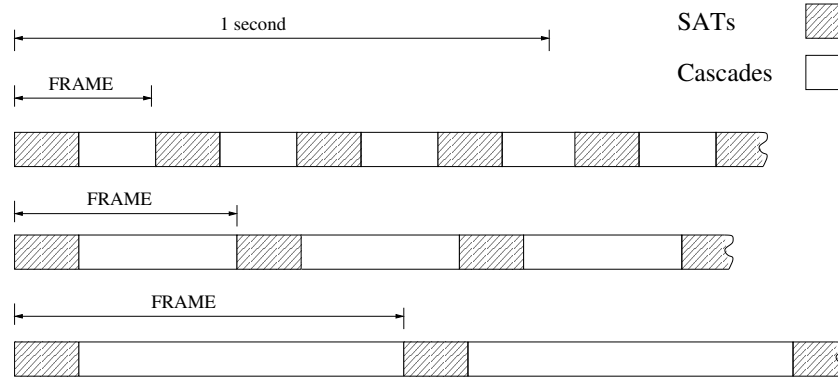


Figure 9.6: Time distribution for concurrent cascades running on one processor.

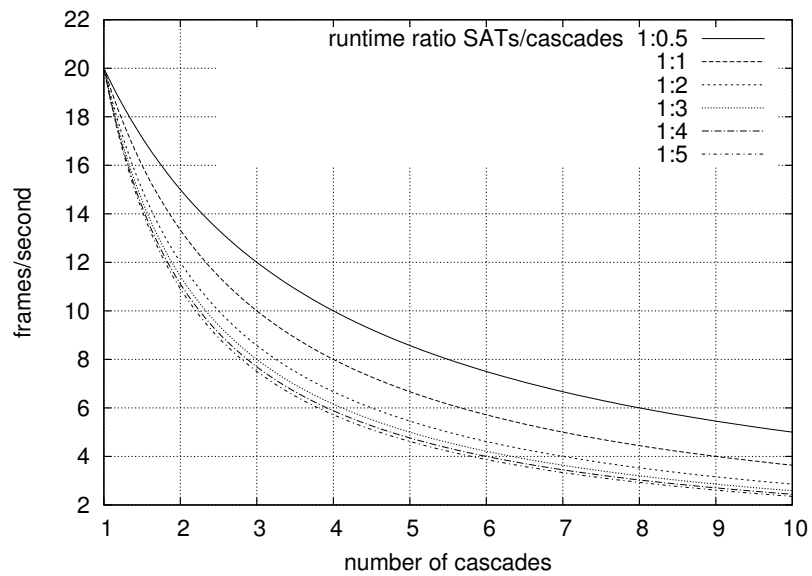


Figure 9.7: Frame rate for concurrent classifiers running on one processor.

has 20 layers, it is trained with a kernel of 24x24 pixels. The figure shows that, in the first layers, a large number of sub-windows are processed with a small number of weak classifiers. In the last layers, a small number of sub-windows are processed by a large number of weak-classifiers. The results fit those presented before in figure 9.4.

A single processor is used to measure the effect of adding cascades to the detector, shown in figure 9.9. The theoretical curve was plotted for the 1:1 ratio for comparison with the theoretical figure 9.7. The frame rate depends only on the ratio of the runtimes for the SAT and the cascades. As the runtime of the cascades are determined by the total number of features computed, the scaling factor S_f and the translation factor T_f can be adjusted to achieve a certain frame rate. However, since both factors influence the accuracy of the cascade a compromise has to be met.

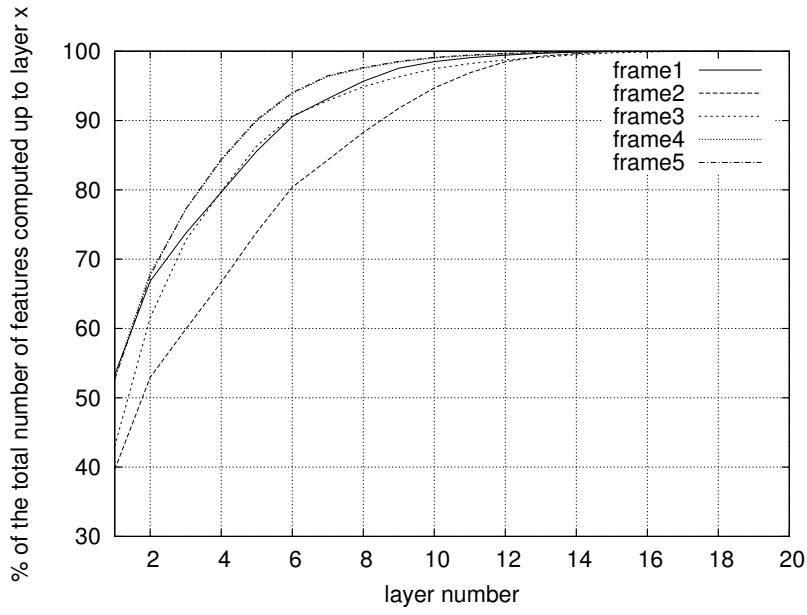


Figure 9.8: Percentage of the total weak classifiers computed up to layer n.

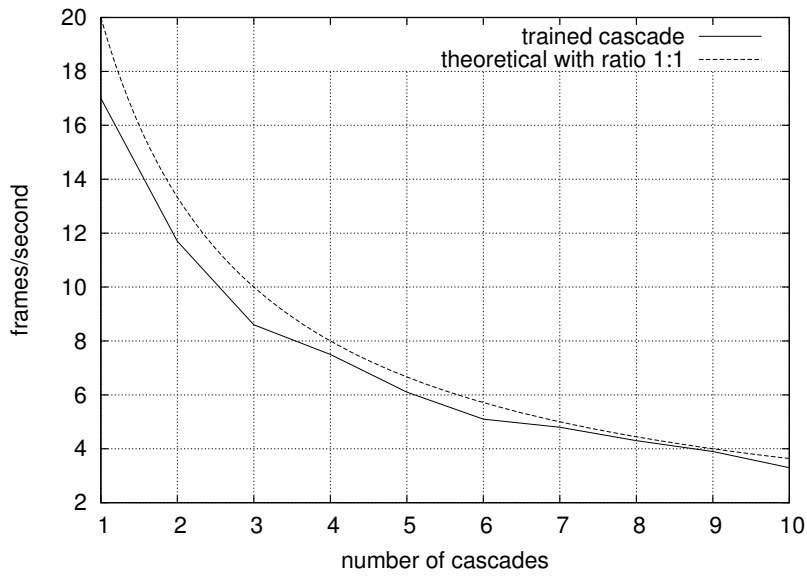


Figure 9.9: Frame rate for concurrent classifiers running on one processor.

9.2.3 Communication

This section briefly discusses the communication issues that arise from using a Beowulf cluster.

The time to send one frame over Ethernet is at least:

$$T_{send} = \frac{8.H.W.\varepsilon}{R} \text{ (seconds)} \quad (9.5)$$

where: H and W are the height and width of the frame in pixels, R is the Ethernet bandwidth in bps and ε is the efficiency of the communication.

In practice, there is a latency time (Wilkinson and Allen, 1999) associated with every data transmission, making every frame broadcast longer than the theoretical limit for a given Ethernet infrastructure. The network protocols themselves can impose an overhead (error correction, layering etc) and the MPI implementation of the broadcast command can vary from platform to platform.

Once a frame is received, each slave node can proceed to compute the SAT. In the case of the original Viola-Jones method, one SAT suffices, as long as only upright Haar-like features are used. However, using other SATs to achieve contrast stretching, to extract tilted Haar-like features or to compute moment invariants, several SATs are necessary (as discussed in chapters 2, 6 and 7). The time to compute n_{SAT} SATs is:

$$T_{compute} = \frac{H.W.n_{SAT}}{P} . f_{SAT} \text{ (seconds)} \quad (9.6)$$

where: P is the processor speed in Hz and f_{SAT} is a correction factor to convert $T_{compute}$ to seconds. The factor f_{SAT} is dependent on the architecture and it estimates the equivalent number of operations needed for the computation of the SATs considering all overheads (such as copying the image to a memory buffer, converting it to grey-scale, and correction factors for the features). The factor f_{SAT} can be estimated by taking the time to compute a SAT using a particular architecture.

Measurements were carried out on two existing Beowulf type clusters at Massey University. The first one is called Sisters and is composed of 8 dual processors connected via a single Gigabit Ethernet switch. The second one is called Helix and is composed of 64 dual processors (most nodes are similar to the Sisters' nodes), but each node has two Gigabit Ethernet adapters. The internal LAN is structured with 7 Gigabit Ethernet switches. A thorough description of the topology and performance of these clusters can be found in section 3.4.

The broadcasting times to send frames to up to 16 nodes from the master node using MPICH¹ were measured. Measurements were repeated three times, regarding the broadcasting of 1000 frames and took the average time to send one frame. Two typical

¹MPICH is an open source implementation of MPI

frame sizes were used: 640x480 and 320x240 pixels. The results are plotted in figure 9.10. The results show that the times vary for different number of slave nodes. The MPICH broadcast is not a true broadcast and it is a blocking call. The broadcast times on Sisters are longer than on Helix because the better quality switches used on Helix makes the latency shorter. Also Helix's nodes have two adapters each, speeding up the communication among a larger number of nodes.

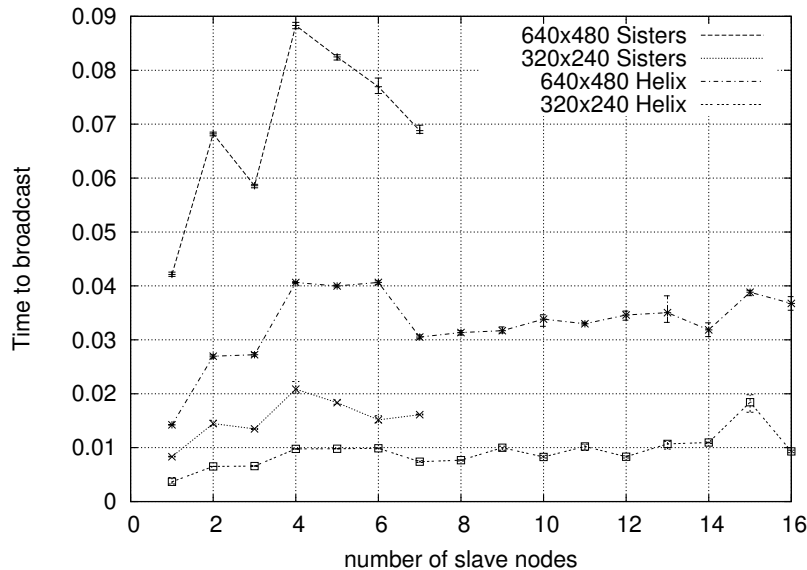


Figure 9.10: Time to broadcast a frame.

9.3 Implementing a mobile cluster using mini-ITX boards

A cluster with four nodes was built based on the proposed platform presented in section 9.1 (figure 9.11). The boards are relatively small (17x17 cm) and have a low power consumption (around 25 watts at full speed). The boards have dual fast Ethernet (100 Mbps), which allows them to use two different communication channels, and each board has 1 CPU (VIA C3/Eden 1000 MHz) with 512 MB RAM. The operating system is a modified Linux (kernel 2.6) with OpenCV library installed.

The communication between the nodes used simple sockets due to the high latency of MPICH shown in figure 9.10. Instead of using the MPI broadcast, a simple protocol based on UDP was used. This allows the implementation of a true broadcast, as opposed to the structured broadcast implemented in MPICH (Gropp, Lusk and Skjellum (1999)).

The time to transfer images is always faster than transferring their SATs (SATs have the same image resolution, but they need more bytes per pixel). If the images are broadcast, the computation of the SATs needs to be repeated in every slave node. Two scenarios may be considered:

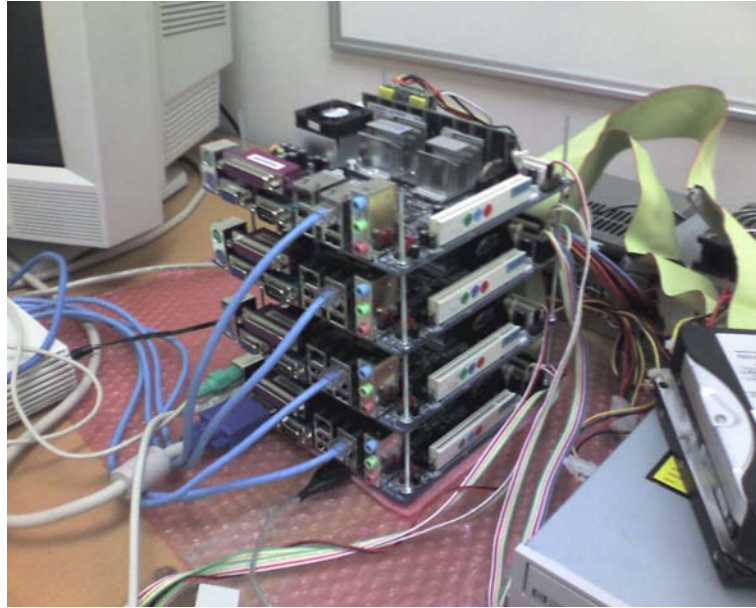


Figure 9.11: A mobile cluster.

Table 9.1: Results for the broadcasting times and SAT runtimes.

Image Size	Image Transfer Time (ms)	SAT Transfer Time (ms)	SAT Runtime (ms)
320x240	11.1	88.7	0.83
640x480	27.4	219.2	3.35
800x600	43.3	346.4	5.26

- **Scenario 1:** Images are broadcast by the master node. The slave nodes compute both the SAT and the cascades.
- **Scenario 2:** The master node computes a SAT and broadcast it to the slave nodes. The slave nodes only run cascades.

A comparison between the image transfer times and the SATs computation times is presented in table 9.1. The results confirm the assumptions made in section 9.2 that it is faster to send images and compute SATs locally, even if that means repeating the SATs computation in each node. Scenario 1 is more suitable to be used by Beowulf clusters.

With the development of faster networks, scenario 2 may become more attractive. However, such communication speeds are not likely to occur without a corresponding improvement in the CPU speed, in which case, scenario 1 is still better suited to the Viola-Jones method. In order to make scenario 2 work for the described mini-cluster, the network has to work at bandwidth rates of at least 10Gbps (about 100 times faster than fast Ethernet). For example, assuming that the frame size is 320x240 pixels, sending a

SAT takes approximately 89 ms. The computation of one SAT for the same size takes only 0.83 ms.

The cascade can demand a lot of the available cputime. Using the face detection from OpenCV, frames with 320x240 pixels can take up to 500 ms, if the scaling factor is 1.1. If larger scaling factors are used, the time can drop. For example, for a scaling factor of 1.6 the cascades runtime drop to about 100 ms . That gives an indication of the limited number of cascades that can run in each node (the 1GHz CPUs are relatively slow compared to CPUs used in desktop computers).

Finally, the position of the detected object has to be sent back to the master. Usually, only a few sub-windows present positive results. The amount of data is so small that the time to send this information is almost the same as the latency. The latency for this mini-cluster was measured at around 6ms², negligible in relation to the cascade runtimes or the broadcasting times.

9.3.1 Speedup

Speedup can be analysed taking into consideration the serial and the parallel parts of the application. If the number of nodes and the percentage of the serial part of an application are known, it is possible to estimate the maximum speedup with Amdahl's Law and with Gustafson's Law (equations 2.10 and 2.11, respectively (Wilkinson and Allen, 1999)).

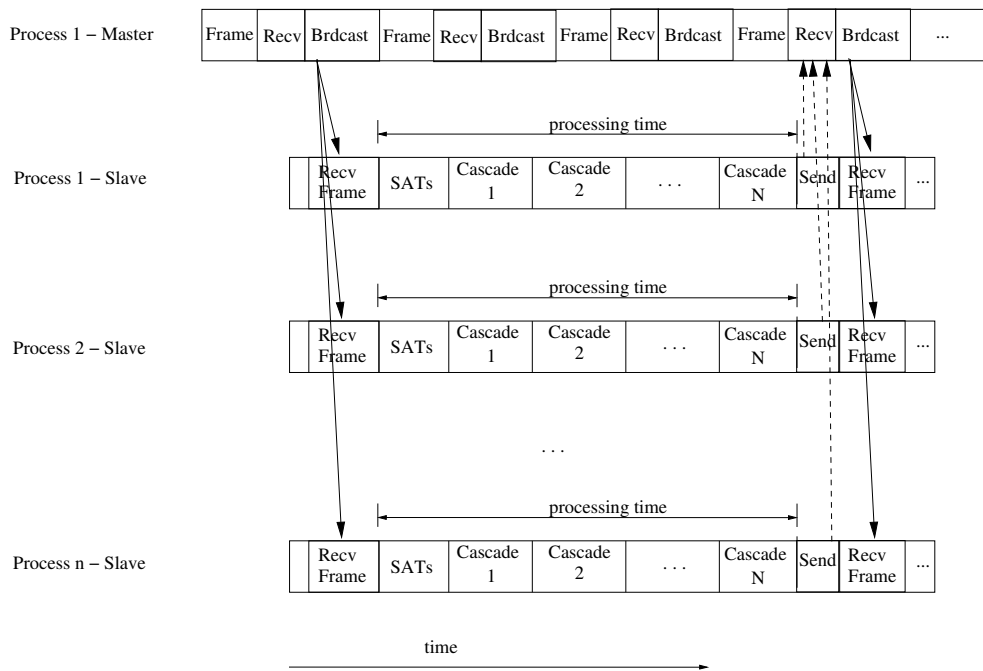


Figure 9.12: Parallelisation scenario: runtime and communication times.

Using the parallelisation model discussed above, the detection task is divided in four

²The mobile cluster uses UDP, rather than MPICH

main parts: broadcasting, SATs, cascades and return of the position of the objects (figure 9.12). The master is assumed to be dedicated to acquire frames via a web camera. The runtimes of each individual part may vary considerably and they can be estimated using the approach presented in the previous section. As an example, suppose that a detection applications needs 32 cascades on a single processor with an image size of 320x240 pixels. Assuming the times discussed in the previous section, the SAT takes 0.83 ms and each cascade takes on average 100 ms (a fair estimate considering that some cascades are faster than others). The total runtime takes $0.83 + 100 * 32 = 3200.83$ ms in one processor.

In order to divide the cascades among two or more nodes, the extra time to make the broadcast and the return of the results needs to be considered. Assuming that the communication times are approximately the ones gathered in the previous section, the image broadcast takes 11.1 ms and the return of results about 6 ms per node. Therefore, the proportion between the serial and the parallel parts are:

serial portion: $0.83 + 11.1 + 6 = 17.93$ ms

parallelisable portion: $100 * 32 = 3200$ ms

According to Amdahl's law (Wilkinson and Allen, 1999), the maximum speedup M_A for 32 nodes is:

$$M_A = \frac{32}{1 + (32 - 1) * (17.93/3217.93)} = 27.31$$

And according to Gustafson's law (Wilkinson and Allen, 1999), the maximum speedup M_G for 32 nodes is:

$$M_G = \frac{17.93}{3217.93} + (1 - \frac{17.93}{3217.93})32 = 31.83$$

This difference in maximum speedup is explained by the fact that the problem size is not very big in relation to the communication times. As the number of cascades increases, the maximum speedup given by Amdahl's law gets closer to the one given by Gustafson's law because the serial portion is fixed for a certain image size.

A minimum of one cascade must run on a given node. Figure 9.13 shows examples with 32, 16, 8 and 4 cascades. The speedup is almost linear, reaching 3.5 for a 4 processors cluster running 4 cascades.

The cascade runtimes are assumed to be evenly distributed among the nodes. However, as seen in figure 9.7, if the cascades are detecting the same objects at different angles, the runtime does not increase linearly. This suggests that the distribution of the cascades among the nodes should be based on specific information about the cascades behaviour or that dynamic load balancing is used.

9.4 Summary

A simple model for the performance of a mobile platform was discussed, based on Beowulf clusters running parallel Viola-Jones detectors. The model predicts the number of nodes

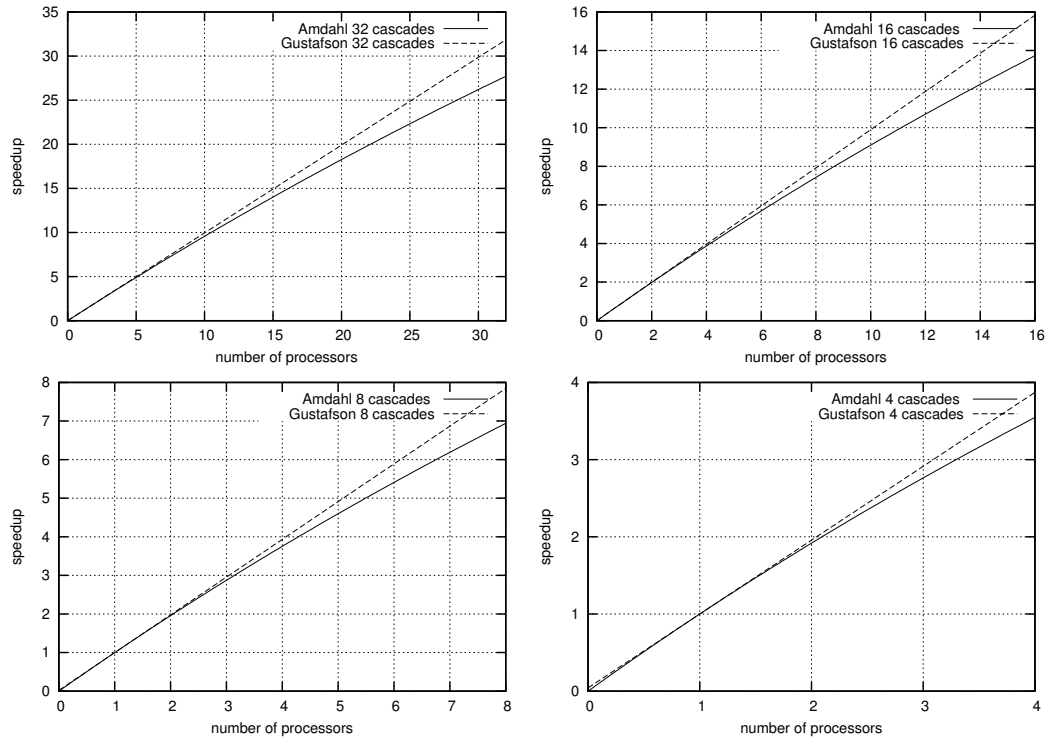


Figure 9.13: Speedups with different number of cascades and processors. The ratio between the parallel and serial parts are derived from the numbers gathered in section 9.3

needed for detection based on Viola-Jones method, based on parameters such as the number of cascades, the number of weak classifiers, the frame size (and scaling factors), and the processing capacity of each node.

The preliminary results fit well with the model. For example, the model explains the performance of running concurrent cascades, each detecting the same object at different angles, on one processor.

The parallelisation strategy for a cluster has to ensure that the communication times are minimised. Function decomposition can be easily implemented, based on the fact that each cascade can work independently, but cascades share one or more SATs. The results showed that if fast Ethernet (100 Mbps) is used, it is faster to broadcast the frames and recompute SATs in each node.

An interesting extension of this work involves dynamic load balancing. It has been shown that the cascades that are not detecting objects tend to run faster than those that are detecting one or more objects. After a few frames, it is possible to determine which cascades are spending more CPU time. Slower cascades can be relocated to nodes that are less occupied, optimising the maximum speedup and achieving a higher frame rate.

Chapter 10

Final Conclusions

10.1 Summary

The main contribution of this thesis is the development of a new approach for real-time detection using Moment Invariants and SATs. The feature extraction method, called CDMI, computes moments from concentric discs of the image. This increases the dimensionality of the moment invariants feature set, without the use of high order moments (this work used low order moments up to the 4th order). This approach allows to enrich the feature set without the usual noise problems related to high order moment invariants. The feature set holds the rotation invariance and scaling invariance properties and the use of SATs allows to extract features from reasonably sized frames very rapidly.

Other original contributions include:

Assessment of the Viola-Jones method for object detection with partial occlusions has been carried out. The experiments showed that random patches on the training images can successfully simulate partial occlusions, extending the range of detection classifiers.

Hand detection using the Viola-Jones method has been analysed. In the training process, modified samples of hand images forced the training to concentrate on the hand's region. The performance of multiple cascades focusing on different angles of rotation was also assessed.

A new rotation approach for Haar-like features, called PEF, has been proposed. In this approach classifiers can be converted to certain angles, with the assistance of special SATs. The values of two features are combined to detect objects at arbitrary angles, without specifically training for those angles. Tilted SATs were used to measure the errors of conversions to 45°. The experiments showed that cascades converted using tilted SATs are accurate enough to yield reliable cascades at angles multiple of 45°, without any extra training. A new SAT computing Haar-like features at angles of 26.5° (and 63.5°) has also been evaluated.

Real-time handwritten digit recognition has been tackled using the new feature extrac-

tion method (CDMI) and its variant (η features using concentric squares). The results, assessed with the help of the MNIST database, were encouraging. The new feature extraction methods allow for fast training, as the dimensionality of the feature set is limited to less than a hundred. Other advantages of the method include fast detection, scaling invariance, and in the case of the CDMI, rotation invariance.

The parallelisation of detection algorithms based on cascade classifiers has been discussed, showing that good speedups can be achieved using a simple frame broadcast to the slave nodes. The fact that SATs can be shared by more than one cascade within the same node makes the parallelisation efficient. A model for a mobile cluster has been proposed in order to estimate the performance, given the number of cascades, the number of nodes, scale factor, translation factor, and the frame resolution. Concurrency can be made efficient for cascades that are classifying the same type of object being detected at different angles, because usually only one cascade has to complete the computation of all its layers.

10.2 Limitations

The training was limited by the simplicity of the implementations of AdaBoost, using thresholds as weak classifiers. The concepts discussed here can be extended by using other variants of the AdaBoost algorithm proposed by Freund and Schapire (1996). For example, the accuracy of the digits classification was, in part, limited by the approach used to combine 90 binary classifiers, instead of using a more sophisticated multiclass approach.

The face classifiers were limited to frontal faces and to the negative samples collected from the web and from other sources (such as the CorelDraw dataset). In the case of the PEF approach, training was limited by the large number of Haar-like features in the training set. The CMU-MIT database was only used to assess the performance of the OpenCV sample cascade converted to angles of 45° . In the case of the CDMI approach, the discrimination powers of the feature set limited the accuracy of the classifiers, so they were not assessed using the CMU-MIT database.

Colour segmentation was not included in the scope of this thesis. However, colour segmentation can improve the accuracy of the hand detection and the face detection by limiting the region where the features are extracted from.

Although we have demonstrated that both the PEF and the CDMI approaches can run in real-time, computational optimisation was not the focus of this work. In the case of CDMI used for face detection, frame rates of less than 1 frame per second were achieved. The implementation of the Haar-like extraction used for the PEFs involves extra operations and therefore is slower than the implementation in OpenCV. These include counting the pixels in the Haar-like features (using lookups to a unit SAT) and verifying the parity of the position of the vertices of the features.

10.3 Future Work

Parts of this work can be extended and used in a number of real-time applications, including gesture recognition, multiple object detection and some special OCR applications. Also, the accuracy of the classifiers shown in this work can be improved by pre-processing the frames in order to speedup the classification. An example is the case of hand detection and face detection, where colour segmentation can be used.

The CDMI method can benefit from further research using other machine learning methods. The modified AdaBoost used in the experiments proved to be robust and converged to low training errors, but it is important to test other methods to improve results. Other forms of AdaBoost (such as parallel and tree cascades (Lienhart, Liang and Kuranov, 2003)) can further examine the accuracy issues related to the CDMI approach. The clusterisation of the samples can make the training process run faster and produce more accurate results. NNs can be used to compare to the ones presented in this thesis. Finally, a study in multi-class training method for the handwritten digit recognition can compare the results obtained with the multiple binary approach adopted in this thesis.

Another area of interest is the use of recently proposed moments such as Chebyshev Moments. The properties of these moments allowed the reconstruction of images to be quite accurate (see for example Mukundan et al. (2001) and Mukundan (2005)), showing that the discrimination powers can be better than Hu's moments for detection applications. At the time of writing, there was not any known method for rapidly computing the Chebyshev moments in the multiresolution approach (scanning sub-windows). A fast method using FPGAs was proposed by Kotoulas and Andreadis (2006), but there is not any known solution similar to SATs, in which the extraction process can gain performance from a pre-computed structure.

Appendices

Appendix A

Implementation of SATs at 26.5°

Computing features for 26.5°

The computation of the features using a special SAT for angles multiple of 26.5° requires a number of cases to be examined. The position of the fourth point needs to be aligned with the other two points, in order to find the correct value for the area. While in the case of the normal and the tilted integral images the solution is trivial, in the 26.5° SATs the position may depend on the parity of the positions of the initial three points (figure A.1).

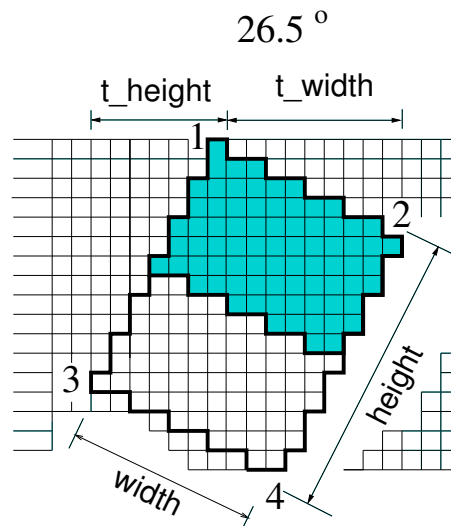


Figure A.1: SAT for 26.5° .

Given four points of the SAT in figure A.1, either the point 4 needs displacement, or both point 4 and point 3 need displacement. If these corrections are not applied, an entire line of pixels could be mistakenly added or subtracted from the intended area. Of the 64 cases, 34 cases do not need any correction, while the remaining 30 cases do. Separate areas need to be treated separately, depending on the feature type.

Given $pt1$, $pt2$, $pt3$, $pt4$ and the width and height of the feature, the 34 cases that do not need displacements are:

$(odd, odd)(odd, odd)(odd, odd)$
 $(odd, odd)(odd, odd)(odd, even)$
 $(odd, odd)(odd, odd)(even, odd)$
 $(odd, odd)(odd, odd)(even, even)$
 $(odd, odd)(odd, even)(odd, odd)$
 $(odd, odd)(odd, even)(even, odd)$
 $(odd, odd)(even, odd)(odd, odd)$
 $(odd, odd)(even, odd)(odd, even)$
 $(odd, odd)(even, even)(odd, odd)$
 $(odd, even)(odd, odd)(odd, even)$
 $(odd, even)(odd, odd)(even, even)$
 $(odd, even)(odd, even)(odd, odd)$
 $(odd, even)(odd, even)(odd, even)$
 $(odd, even)(odd, even)(even, odd)$
 $(odd, even)(odd, even)(even, even)$
 $(odd, even)(even, odd)(odd, even)$
 $(odd, even)(even, even)(odd, odd)$
 $(odd, even)(even, even)(odd, even)$
 $(even, odd)(odd, odd)(even, odd)$
 $(even, odd)(odd, odd)(even, even)$
 $(even, odd)(even, odd)(odd, odd)$
 $(even, odd)(even, odd)(odd, even)$
 $(even, odd)(even, odd)(even, odd)$
 $(even, odd)(even, odd)(even, even)$
 $(even, odd)(even, even)(even, odd)$
 $(even, even)(odd, odd)(even, even)$
 $(even, even)(odd, even)(even, odd)$
 $(even, even)(odd, even)(even, even)$
 $(even, even)(even, odd)(odd, even)$
 $(even, even)(even, odd)(even, even)$
 $(even, even)(even, even)(odd, odd)$
 $(even, even)(even, even)(odd, even)$
 $(even, even)(even, even)(even, odd)$
 $(even, even)(even, even)(even, even)$

For these 34 cases, point 4 is:

$$pt4x = pt3x + width$$

$$pt4y = pt2y + heigh$$

The other 30 cases are:

$$(odd, odd)(odd, even)(odd, even)$$

$$pt3y = pt3y - 1;$$

$$pt4x = pt3x + width;$$

$$pt4y = pt2y + height - 1;$$

$$(odd, odd)(odd, even)(even, even)$$

$$pt4x = pt3x + width - 1;$$

$$pt4y = pt2y + height;$$

$$(odd, odd)(even, odd)(even, odd)$$

$$pt4x = pt3x + width;$$

$$pt4y = pt2y + height + 1;$$

$$(odd, odd)(even, odd)(even, even)$$

$$pt3y = pt3y - 1;$$

$$pt4x = pt3x + width;$$

$$pt4y = pt2y + height;$$

$$(odd, odd)(even, even)(odd, even)$$

$$pt4x = pt3x + width - 1;$$

$$pt4y = pt2y + height;$$

$$(odd, odd)(even, even)(even, odd)$$

$$pt2x = pt2x - 1;$$

$$pt4x = pt3x + width - 1;$$

$$pt4y = pt2y + height;$$

$$(odd, odd)(even, even)(even, even)$$

$$pt4x = pt3x + width - 1;$$

$$pt4y = pt2y + height;$$

$$(odd, even)(odd, odd)(odd, odd)$$

$$pt4x = pt3x + width + 1;$$

$$pt4y = pt2y + height;$$

$(odd, even)(odd, odd)(even, odd)$

$$pt3y = pt3y + 1;$$

$$pt4x = pt3x + width;$$

$$pt4y = pt2y + height + 1;$$

$(odd, even)(even, odd)(odd, odd)$

$$pt3y = pt3y + 1;$$

$$pt4x = pt3x + width;$$

$$pt4y = pt2y + height + 1;$$

$(odd, even)(even, odd)(even, odd)$

$$pt4x = pt3x + width;$$

$$pt4y = pt2y + height + 1;$$

$(odd, even)(even, odd)(even, even)$

$$pt4x = pt3x + width;$$

$$pt4y = pt2y + height + 1;$$

$(odd, even)(even, even)(even, odd)$

$$pt4x = pt3x + width;$$

$$pt4y = pt2y + height + 1;$$

$(odd, even)(even, even)(even, even)$

$$pt3y = pt3y - 1;$$

$$pt4x = pt3x + width;$$

$$pt4y = pt2y + height;$$

$(even, odd)(odd, odd)(odd, odd)$

$$pt3y = pt3y + 1;$$

$$pt4x = pt3x + width;$$

$$pt4y = pt2y + height;$$

$(even, odd)(odd, odd)(odd, even)$

$$pt4x = pt3x + width;$$

$$pt4y = pt2y + height - 1;$$

$(even, odd)(odd, even)(odd, odd)$

$$pt4x = pt3x + width;$$

$$pt4y = pt2y + height - 1;$$

$$(even, odd)(odd, even)(odd, even)$$

$$pt4x = pt3x + width;$$

$$pt4y = pt2y + height - 1;$$

$$(even, odd)(odd, even)(even, odd)$$

$$pt4x = pt3x + width;$$

$$pt4y = pt2y + height;$$

$$(even, odd)(odd, even)(even, even)$$

$$pt3y = pt3y - 1;$$

$$pt4x = pt3x + width;$$

$$pt4y = pt2y + height - 1;$$

$$(even, odd)(even, even)(odd, even)$$

$$pt3y = pt3y - 1;$$

$$pt4x = pt3x + width;$$

$$pt4y = pt2y + height - 1;$$

$$(even, odd)(even, even)(even, even)$$

$$pt4x = pt3x + width - 1;$$

$$pt4y = pt2y + height;$$

$$(even, even)(odd, odd)(odd, odd)$$

$$pt4x = pt3x + width + 1;$$

$$pt4y = pt2y + height;$$

$$(even, even)(odd, odd)(odd, even)$$

$$pt3y = pt3y - 1;$$

$$pt4x = pt3x + width + 1;$$

$$pt4y = pt2y + height - 1;$$

$$(even, even)(odd, odd)(even, odd)$$

$$pt4x = pt3x + width + 1;$$

$$pt4y = pt2y + height;$$

(even, even)(odd, even)(odd, odd)

$pt3y = pt3y + 1;$

$pt4x = pt3x + width;$

$pt4y = pt2y + height;$

(even, even)(odd, even)(odd, even)

$pt4x = pt3x + width;$

$pt4y = pt2y + height - 1;$

(even, even)(even, odd)(odd, odd)

$pt4x = pt3x + width + 1;$

$pt4y = pt2y + height;$

(even, even)(even, odd)(even, odd)

$pt3y = pt3y + 1;$

$pt4x = pt3x + width;$

$pt4y = pt2y + height + 1;$

Appendix B

Moments with SATs

B.1 Deriving The Equations For μ_{pq} From SATs (m_{pq})

For order 1,1 μ_{11} can be derived as follows:

$$\begin{aligned}
 \mu_{11} &= \sum_x \sum_y (x - \bar{x})^1 (y - \bar{y})^1 f(x, y) \\
 &= \sum_x \sum_y (xy - x\bar{y} - \bar{x}y + \bar{x}\bar{y}) f(x, y) \\
 &= \sum_x \sum_y xy f(x, y) - \sum_x \sum_y x\bar{y} f(x, y) - \sum_x \sum_y \bar{x}y f(x, y) + \sum_x \sum_y \bar{x}\bar{y} f(x, y)
 \end{aligned}$$

Both \bar{x} and \bar{y} are constant for a sub-window. Also each of the four factors can be expressed as a function of the corresponding Summed-area Table:

$$\begin{aligned}
 \mu_{11} &= m_{11} - \bar{y}m_{10} - \bar{x}m_{01} + \bar{x}\bar{y}m_{00} \\
 &= m_{11} - \frac{m_{01}m_{10}}{m_{00}} - \frac{m_{10}m_{01}}{m_{00}} + \frac{m_{01}m_{10}m_{00}}{m_{00}^2} \\
 &= m_{11} - \frac{m_{01}m_{10}}{m_{00}}
 \end{aligned} \tag{B.1}$$

Analogously, the other μ_{pq} are derived:

For μ_{30} and μ_{03} :

$$\begin{aligned}
\mu_{30} &= \sum_x \sum_y (x - \bar{x})^3 (y - \bar{y})^0 f(x, y) \\
&= \sum_x \sum_y (x^3 - \bar{x}x^2 + \bar{x}^2x - \bar{x}^3 - 2x^2\bar{x} + 2x\bar{x}^2) f(x, y) \\
&= \sum_x \sum_y (x^3 - 3\bar{x}x^2 + 3\bar{x}^2x - \bar{x}^3) f(x, y) \\
&= m_{30} - 3\bar{x}m_{20} + 3\bar{x}^2m_{10} - \bar{x}^3m_{00} \\
&= m_{30} - \frac{3m_{10}m_{20}}{m_{00}} + \frac{3m_{10}^2m_{10}}{m_{00}^2} - \frac{m_{10}^3}{m_{00}^2} \\
\mu_{30} &= m_{30} - \frac{3m_{10}m_{20}}{m_{00}} + \frac{2m_{10}^3}{m_{00}^2}
\end{aligned} \tag{B.2}$$

and by symmetry, the equation for μ_{03} is:

$$\mu_{03} = m_{03} - \frac{3m_{01}m_{02}}{m_{00}} + \frac{2m_{01}^3}{m_{00}^2} \tag{B.3}$$

For μ_{31} and μ_{13} :

$$\begin{aligned}
\mu_{31} &= \sum_x \sum_y (x - \bar{x})^3 (y - \bar{y})^1 f(x, y) \\
\mu_{31} &= \sum_x \sum_y (x^2 - 2x\bar{x} + \bar{x}^2)(xy - x\bar{y} - \bar{x}y + \bar{x}\bar{y}) f(x, y) \\
\mu_{31} &= \sum_x \sum_y (x^2 - 2x\bar{x} + \bar{x}^2)(xy - x\bar{y} - \bar{x}y + \bar{x}\bar{y}) f(x, y) \\
\mu_{31} &= \sum_x \sum_y (x^3y - x^3\bar{y} - x^2y\bar{x} + x^2\bar{x}\bar{y} - 2x^2y\bar{x} + 2x^2\bar{x}\bar{y} + 2xy\bar{x}^2 \\
&\quad - 2x\bar{x}^2\bar{y} + xy\bar{x}^2 - x\bar{x}^2 - y\bar{x}^3 + \bar{x}^3\bar{y}) f(x, y) \\
\mu_{31} &= m_{31} - m_{30}\bar{y} - m_{21}\bar{x} + m_{20}\bar{x}\bar{y} - 2m_{21}\bar{x} + 2m_{20}\bar{x}\bar{y} \\
&\quad + 2m_{11}\bar{x}^2 - 2m_{10}\bar{x}^2\bar{y} + m_{11}\bar{x}^2 - m_{10}\bar{x}^2\bar{y} - m_{01}\bar{x}^3 + \bar{x}^3\bar{y}
\end{aligned}$$

$$\mu_{31} = m_{31} - \bar{y}m_{30} + 3\bar{x}\bar{y}(m_{20} - m_{21}) + 3\bar{x}^2(m_{11} - \bar{y}m_{10}) + \bar{x}^3(\bar{y}m_{00} - m_{01}) \quad (\text{B.4})$$

and by symmetry, μ_{13} is:

$$\mu_{13} = m_{13} - \bar{x}m_{03} + 3\bar{x}\bar{y}(m_{02} - m_{12}) + 3\bar{y}^2(m_{11} - \bar{x}m_{01}) + \bar{y}^3(\bar{x}m_{00} - m_{10}) \quad (\text{B.5})$$

For μ_{40} and μ_{04} :

$$\begin{aligned} \mu_{40} &= \sum_x \sum_y (x - \bar{x})^4 (y - \bar{y})^0 f(x, y) \\ \mu_{40} &= \sum_x \sum_y (x^2 - 2x\bar{x} + \bar{x}^2)^2 f(x, y) \\ \mu_{40} &= \sum_x \sum_y (x^4 - 4x^3\bar{x} + 6x^2\bar{x}^2 - 4x\bar{x}^3 + \bar{x}^4) f(x, y) \\ \mu_{40} &= m_{40} - 4\bar{x}m_{30} + 6\bar{x}^2m_{20} - 4\bar{x}^3m_{10} + \bar{x}^4m_{00} \end{aligned} \quad (\text{B.6})$$

and by symmetry, μ_{04} is:

$$\mu_{04} = m_{04} - 4\bar{y}m_{03} + 6\bar{y}^2m_{02} - 4\bar{y}^3m_{01} + \bar{y}^4m_{00} \quad (\text{B.7})$$

B.2 4th Order Moment Invariants Derivation

The formulae used to compute 3rd and 4th order moment invariants were not explicitly derived in Flusser (2000a). Using the notation developed by Flusser, the independent set of moment invariants up to the 4th order is:

2rd order:

$$\psi_1 = c_{11} = \phi_1$$

3rd order:

$$\psi_2 = c_{21}c_{12} = \phi_4$$

$$\psi_3 = \text{Re}(c_{20}c_{12}^2) = \phi_6$$

$$\psi_4 = \text{Im}(c_{20}c_{12}^2)$$

$$\psi_5 = \text{Re}(c_{30}c_{12}^3) = \phi_5$$

$$\psi_6 = \text{Im}(c_{30}c_{12}^3) = \phi_7$$

4th order:

$$\psi_7 = c_{22}$$

$$\psi_8 = \text{Re}(c_{31}c_{12}^2)$$

$$\psi_9 = \text{Im}(c_{31}c_{12}^2)$$

$$\psi_{10} = \text{Re}(c_{40}c_{12}^4)$$

$$\psi_{11} = \text{Im}(c_{40}c_{12}^4)$$

Where ψ_n are Flusser's moments, ϕ_n are Hu's moments, and

$$c_{pq} = (x + yi)^p(x - yi)^q f(x, y) \text{ with } i = \sqrt{-1}.$$

The moments $\psi_1, \psi_2, \psi_3, \psi_5$ and ψ_6 correspond to 5 independent Hu's moments (Hu, 1962). Flusser (2000a) wrote ψ_4 explicitly in terms of η_{pq} . The remaining five 4th order moments' are derived next.

$$\begin{aligned} \psi_7 = c_{22} &= \sum_x \sum_y (x + iy)_2(x - iy)_2 f(x, y) \\ &= \sum_x \sum_y (x^2 + y^2i^2 + 2xyi)(x^2 + y^2i^2 - 2xyi) f(x, y) \\ &= \sum_x \sum_y (x^4 - x^2y^2i^2 - 2x^3yi - x^2y^2 + y^4 + 2xy^3i + 2x^3yi - 2xy^3i + 4x^2y^2) f(x, y) \end{aligned}$$

$$\begin{aligned}
&= \sum_x \sum_y (x^4 + y^4 + 2x^2y^2)f(x, y) \\
&= \sum_x \sum_y x^4 f(x, y) + \sum_x \sum_y y^4 f(x, y) + \sum_x \sum_y 2x^2y^2 f(x, y)
\end{aligned}$$

$$\psi_7 = \eta_{40} + \eta_{04} + 2\eta_{22}$$

(B.8)

For ψ_8 and ψ_9 , one needs to compute $c_{31}c_{12}^2$ and split the real and imaginary parts:

$$c_{31} = \sum_x \sum_y (x + yi)^3 (x - yi) f(x, y)$$

$$c_{31} = \sum_x \sum_y (x^2 + y^2i^2 + 2xyi)(x + yi)(x - yi) f(x, y)$$

$$c_{31} = \sum_x \sum_y (x^2 + y^2i^2 + 2xyi)(x^2 - y^2i^2) f(x, y)$$

$$c_{31} = \sum_x \sum_y (x^4 - x^2y^2i^2 + x^2y^2i^2 - y^4i^4 + 2x^3yi - 2xy^3i^3) f(x, y)$$

$$c_{31} = \sum_x \sum_y (x^4 - y^4 + 2x^3yi + 2xy^3i) f(x, y)$$

$$c_{31} = \eta_{40} - \eta_{04} + 2(\eta_{31} + \eta_{13})i$$

$$c_{12} = \sum_x \sum_y (x + yi)(x - yi)^2 f(x, y)$$

$$c_{12} = \sum_x \sum_y (x + yi)(x^2 + y^2i^2 - 2xyi) f(x, y)$$

$$c_{12} = \sum_x \sum_y (x^3 + xy^2i^2 - 2x^2yi + x^2yi + y^3i^3 - 2xy^2i^2) f(x, y)$$

$$c_{12} = \sum_x \sum_y (x^3 - xy^2 - x^2yi + y^3i^3 + 2xy^2) f(x, y)$$

$$c_{12} = \sum_x \sum_y (x^3 + xy^2 - x^2yi - y^3i) f(x, y)$$

$$c_{12} = \eta_{30} + \eta_{12} - (\eta_{21} + \eta_{03})i$$

The expression $c_{31}c_{12}^2$ can be written as:

$$c_{31}c_{12}^2 = [\eta_{40} - \eta_{04} + 2(\eta_{31} + \eta_{13})i][\eta_{30} + \eta_{12} - (\eta_{21} + \eta_{03})i]^2$$

$$c_{31}c_{12}^2 = [\eta_{40} - \eta_{04} + 2(\eta_{31} + \eta_{13})i][(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2 - 2(\eta_{30} + \eta_{12})(\eta_{21} + \eta_{03})i]$$

$$c_{31}c_{12}^2 = (\eta_{40} - \eta_{04})[(\eta_{30} + \eta_{12})^2 - (\eta_{03} + \eta_{21})^2] - 4(\eta_{31} + \eta_{13})(\eta_{30} + \eta_{12})(\eta_{03} + \eta_{21})i^2 + 2(\eta_{31} + \eta_{13})[(\eta_{30} + \eta_{12})^2 - (\eta_{03} + \eta_{21})^2]i - 2(\eta_{40} - \eta_{04})(\eta_{30} + \eta_{12})(\eta_{03} + \eta_{21})i$$

Splitting the real and imaginary components, ψ_8 and ψ_9 can be expressed as:

$$\psi_8 = (\eta_{40} - \eta_{04})[(\eta_{30} + \eta_{12})^2 - (\eta_{03} + \eta_{21})^2] + 4(\eta_{31} + \eta_{13})(\eta_{30} + \eta_{12})(\eta_{03} + \eta_{21}) \quad (\text{B.9})$$

$$\psi_9 = 2(\eta_{31} + \eta_{13})[(\eta_{30} + \eta_{12})^2 - (\eta_{03} + \eta_{21})^2] - 2(\eta_{40} - \eta_{04})(\eta_{30} + \eta_{12})(\eta_{21} + \eta_{03}) \quad (\text{B.10})$$

The two final moments, ψ_{10} and ψ_{11} , need c_{40} and c_{12}^4 :

$$c_{40} = \sum_x \sum_y (x + yi)^4 f(x, y) = \sum_x \sum_y (x^2 + y^2i^2 + 2xyi)^2 f(x, y)$$

$$c_{40} = \sum_x \sum_y (x^4 + x^2y^2i^2 + 2x^3yi + x^2y^2i^2 + y^4i^4 + 2xy^3i^3 + 2x^3yi + 2xy^3i^3 + 4x^2y^2i^2) f(x, y)$$

$$c_{40} = \sum_x \sum_y (x^4 + y^4 - 6x^2y^2 + 4x^3yi - 4xy^3i) f(x, y)$$

$$c_{40} = (\eta_{40} + \eta_{04} - 6\eta_{22}) + 4(\eta_{31} - \eta_{13})i$$

$$c_{12} = \sum_x \sum_y (x + yi)(x - yi)^2 f(x, y) = \sum_x \sum_y (x + yi)(x^2 + y^2i^2 - 2xyi)^2 f(x, y)$$

$$c_{12} = \sum_x \sum_y [(x^3 + xy^2) - (x^2y + y^3)i] f(x, y)$$

$$c_{12} = (\eta_{30} + \eta_{12}) - (\eta_{21} + \eta_{03})i$$

$$c_{12}^4 = [(\eta_{30} + \eta_{12})^2 + (\eta_{21} + \eta_{03})^2 i^2 - 2(\eta_{30} + \eta_{12})(\eta_{21} + \eta_{03})]^2$$

$$c_{12}^4 = (\eta_{30} + \eta_{12})^4 + (\eta_{21} + \eta_{03})^4 i^4 - 6(\eta_{30} + \eta_{12})^2(\eta_{21} + \eta_{03})^2 \\ - 4(\eta_{30} + \eta_{12})^3(\eta_{03} + \eta_{21})i + 4(\eta_{30} + \eta_{12})(\eta_{03} + \eta_{21})^3 i$$

$$c_{12}^4 = (\eta_{30} + \eta_{12})^4 + (\eta_{21} + \eta_{03})^4 i^4 - 6(\eta_{30} + \eta_{12})^2(\eta_{21} + \eta_{03})^2 \\ - 4[(\eta_{30} + \eta_{12})^3(\eta_{03} + \eta_{21})i - (\eta_{30} + \eta_{12})(\eta_{03} + \eta_{21})^3 i]$$

$Re(c_{40}c_{12}^4)$ (ψ_{10}) is:

$$\psi_{10} = (\eta_{40} - 6\eta_{22} + \eta_{04})\{[(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2]^2 - 4(\eta_{30} + \eta_{12})^2(\eta_{03} + \eta_{21})^2\} \\ + 16(\eta_{31} - \eta_{13})(\eta_{30} + \eta_{12})(\eta_{03} + \eta_{21})[(\eta_{30} + \eta_{12})^2 - (\eta_{03} + \eta_{21})^2] \quad (B.11)$$

And $Im(c_{40}c_{12}^4)$ (ψ_{11}) is:

$$\psi_{11} = 4(\eta_{40} - 6\eta_{22} + \eta_{04})(\eta_{30} + \eta_{12})(\eta_{03} + \eta_{21})[(\eta_{30} + \eta_{12})^2 - (\eta_{03} + \eta_{21})^2] \\ - 4(\eta_{31} - \eta_{13})\{[(\eta_{30} + \eta_{12})^2 - (\eta_{03} + \eta_{21})^2]^2 - 4(\eta_{30} + \eta_{12})^2(\eta_{03} + \eta_{21})^2\} \quad (B.12)$$

Appendix C

Additional Results for Digits Recognition

This appendix includes tables that, due to space constraints, could not be presented in chapter 8. The test errors were carried out based on the 10000 MNIST dataset.

C.1 Experiment 1: additional Tables

The false detection results for CH-11- ψ are shown in table C.1. This table was created running each CH-11- ψ classifier against all the other classes.

Table C.1: CH-11- ψ : false detection rate (%) per classifier.

Classifier	Negative class									
	0	1	2	3	4	5	6	7	8	9
0	-	7.0	52.7	25.8	15.3	34.6	16.4	15.1	28.2	28.0
1	1.4	-	26.5	27.5	19.3	56.5	9.5	19.9	26.7	21.8
2	42.1	32.7	-	81.7	78.7	86.8	79.7	77.8	79.7	84.2
3	49.0	50.5	92.1	-	77.8	91.0	83.9	81.1	82.0	89.6
4	16.6	28.9	78.0	70.1	-	75.7	75.3	65.9	77.7	83.8
5	50.3	52.9	88.6	82.7	70.6	-	72.0	73.3	79.5	82.3
6	6.2	7.5	51.9	57.0	54.0	48.4	-	64.1	38.9	85.3
7	5.1	13.8	75.0	64.3	56.3	73.3	80.7	-	38.8	73.9
8	66.5	55.5	90.2	85.0	80.5	87.7	63.9	61.9	-	82.2
9	20.3	31.4	71.0	76.5	83.1	72.1	91.3	75.6	71.6	-

The false detection results for CH-88- ψ are shown in table C.2. This table was created running each CH-88- ψ classifier against all the other classes.

The overall confusion matrix for Ada-11- ψ -50 is shown in table C.3.

The overall confusion matrix for Ada-88- ψ -50 is shown in table C.4.

Table C.2: CH-88- ψ : false detection rate (%) per classifier.

Classifier	Negative class									
	0	1	2	3	4	5	6	7	8	9
0	-	0.1	15.9	14.0	7.9	18.7	3.1	2.7	23.2	10.4
1	0.1	-	2.5	2.6	1.2	4.5	0.4	0.7	12.3	0.7
2	13.8	4.4	-	51.1	51.4	56.1	32.2	40.5	50.8	38.0
3	4.2	9.9	40.8	-	33.2	46.0	29.2	25.2	49.8	26.6
4	3.7	0.6	22.4	18.6	-	22.9	23.1	15.6	37.9	45.6
5	6.7	10.8	47.6	43.7	41.8	-	26.0	35.4	48.2	32.0
6	4.9	1.1	33.9	34.1	42.1	32.1	-	45.2	29.2	44.2
7	4.0	2.3	39.0	30.8	32.9	36.5	41.0	-	23.9	29.5
8	1.9	1.5	15.2	22.4	31.2	21.0	11.1	5.2	-	29.9
9	5.8	2.2	24.7	29.1	52.5	28.9	49.7	25.6	47.1	-

Table C.3: Overall Confusion Matrix for Ada-11- ψ -50.

	predicted									
	0	1	2	3	4	5	6	7	8	9
Actual class 0	50.1	0.51	4.69	6.53	0.71	11.12	0	0.51	24.9	0.92
1	0.18	61.67	0.18	2.2	0.35	11.54	0	1.85	20.44	1.59
2	2.03	0.48	23.84	13.28	8.62	7.66	3.49	14.92	17.64	8.04
3	0.5	0.5	11.49	36.14	1.09	8.91	3.66	14.46	17.03	6.24
4	1.22	0.41	8.55	2.04	26.99	3.67	1.63	9.27	26.17	20.06
5	3.03	1.68	8.86	14.69	2.35	19.84	0.9	18.5	23.32	6.84
6	0.31	0	4.91	3.03	3.03	0.73	44.89	20.15	3.55	19.42
7	0.19	0.1	4.38	4.09	1.56	1.17	8.17	67.02	2.33	10.99
8	5.24	0.51	6.78	8.73	2.98	7.6	0.41	2.05	61.7	4
9	0.89	0.1	6.34	2.58	5.65	3.07	14.77	11.4	12.88	42.32

Table C.4: Overall Confusion Matrix for Ada-88- ψ -50.

	predicted									
	0	1	2	3	4	5	6	7	8	9
Actual class 0	82.76	0	5.92	0.92	0.51	4.08	0.61	0.41	2.35	2.45
1	0	90.75	0.35	2.38	0.09	2.38	0	0.26	3.7	0.09
2	3.39	0.87	51.74	5.62	7.46	11.34	2.42	7.66	7.07	2.42
3	0.3	0.1	11.78	54.16	2.08	10.3	2.57	3.27	13.07	2.38
4	0.31	0.1	12.32	1.73	44.91	6.62	2.95	6.11	14.46	10.49
5	2.13	0.45	20.52	7.51	3.59	36.66	1.79	12.11	10.09	5.16
6	0.73	0.1	6.26	3.24	2.71	1.04	57.41	14.82	2.09	11.59
7	0.29	0.68	6.03	2.53	2.53	2.92	10.31	67.7	0.78	6.23
8	0.51	0.51	3.59	9.14	3.7	6.26	0.62	0.82	70.43	4.41
9	1.68	0.2	2.08	1.39	5.85	3.67	10.01	4.86	3.27	67

C.2 Experiment 2: additional Tables

These results were collected by running the classifiers individually, using positive and negative images for which the classifier was trained. The hit rates for ψ -Ada-88-N classifiers are shown in table C.5. The false detection rates for ψ -Ada-88-N classifiers are shown in table C.6.

Table C.5: ψ -Ada-88-N: hit rate (%) per classifier.

Classifier	Negative class									
	0	1	2	3	4	5	6	7	8	9
0	-	99.6	95.7	98.6	98.2	97.4	98.9	98.7	98.7	98.2
1	99.7	-	99.0	98.7	99.0	98.5	99.5	99.1	98.7	99.4
2	96.2	98.2	-	90.5	89.1	87.5	95.3	92.0	91.6	94.7
3	98.5	98.8	90.9	-	93.5	89.2	95.5	95.6	90.5	95.8
4	97.6	99.1	87.9	93.7	-	93.2	92.6	91.4	89.9	89.5
5	95.4	98.2	81.5	85.0	87.0	-	93.7	90.6	88.7	94.0
6	98.1	99.3	93.5	94.9	94.7	94.9	-	91.6	97.3	93.8
7	99.0	98.7	91.5	94.4	93.8	93.1	91.3	-	97.2	92.8
8	96.9	98.7	93.1	89.0	92.2	91.6	96.1	97.0	-	95.9
9	98.2	99.1	94.9	95.5	92.1	95.1	93.1	93.2	97.2	-

Table C.6: ψ -Ada-88-N: false detection rate (%).

Classifier	Negative class									
	0	1	2	3	4	5	6	7	8	9
0	-	1.2	18.2	8.9	11.3	22.8	5.6	3.7	15.6	8.0
1	1.5	-	9.3	9.4	7.3	13.2	3.5	4.8	9.3	4.9
2	14.7	4.0	-	35.2	39.4	51.2	24.4	26.4	29.4	22.4
3	6.3	4.6	33.4	-	29.4	46.0	20.5	17.1	35.5	18.2
4	7.2	4.3	34.1	22.5	-	37.1	17.6	19.4	28.5	30.6
5	11.3	04.9	43.5	35.5	33.9	-	18.7	23.6	27.3	20.3
6	4.5	2.4	22.6	19.0	27.0	24.6	-	27.9	16.0	28.4
7	5.1	2.9	29.2	20.4	31.7	31.7	26.5	-	16.3	24.4
8	6.7	5.9	21.9	28.6	34.8	29.2	10.8	10.1	-	18.7
9	10.5	2.7	22.5	18.1	36.4	28.0	28.0	22.6	25.3	-

C.3 Experiment 3: additional Tables

These results were collected by running the classifiers individually, using positive and negative images for which the classifier was trained. The hit rates for $\eta - 84 - N$ classifiers are shown in table C.7. The false detection rates for $\eta - 84 - N$ classifiers are shown in table C.8. When two or more get a hit with a certain image, a choice has been made using the criteria explained in chapter 8. Table C.9 presents the errors caused by wrong classifications when faced with a choice.

Table C.7: $\eta - 84 - N$: hit rate (%) per classifier.

Classifier	Negative class									
	0	1	2	3	4	5	6	7	8	9
0	-	100.0	99.5	99.5	99.8	99.1	99.5	99.5	99.5	99.7
1	99.9	-	99.6	99.6	99.9	99.6	100.0	99.9	99.8	99.9
2	98.6	99.8	-	97.6	99.3	98.8	98.9	99.0	98.2	99.2
3	99.8	99.7	98.9	-	99.3	98.9	99.8	98.9	98.6	98.1
4	99.6	100.0	98.9	99.3	-	98.6	98.4	99.3	98.8	97.2
5	98.8	99.5	99.3	96.7	98.5	-	99.3	99.1	96.8	99.1
6	98.6	99.3	98.6	98.9	98.7	98.6	-	99.3	99.1	99.5
7	99.3	99.3	98.4	98.5	98.8	99.2	99.4	-	99.3	97.3
8	99.5	99.9	99.3	98.3	98.0	97.8	99.8	99.2	-	98.6
9	99.7	99.2	99.3	98.8	97.0	99.0	99.7	98.3	98.4	-

Table C.8: $\eta - 84 - N$: false detection rate (%).

Classifier	Negative class									
	0	1	2	3	4	5	6	7	8	9
0	-	0.44	2.71	0.69	1.73	2.91	1.98	0.87	1.74	0.89
1	0.00	-	2.51	0.89	0.91	1.23	0.73	1.65	1.23	0.89
2	1.02	1.93	-	4.45	4.37	2.91	4.59	4.37	4.62	3.07
3	0.91	1.76	6.87	-	2.64	7.95	1.56	5.44	9.24	4.06
4	1.32	0.17	4.65	2.27	-	3.81	3.65	4.47	7.39	15.06
5	2.24	0.79	3.19	4.65	3.46	-	3.75	1.07	7.28	2.77
6	1.53	0.52	4.06	0.99	3.25	3.13	-	1.07	3.18	0.59
7	0.91	0.88	3.77	3.06	4.07	2.57	0.20	-	2.87	6.73
8	1.02	1.85	4.94	4.75	5.29	8.74	2.19	2.82	-	5.25
9	0.81	0.35	3.48	3.56	13.03	2.80	0.41	8.46	5.54	-

Bibliography

- Akl, S. and Toussaint, G. (1978), Efficient convex hull algorithms for pattern recognition applications, *in* 'Proc. 4th Int'l Joint Conf. on Pattern Recognition', Kyoto, Japan, pp. 483–487.
- Andrew, A. (1979), "Another efficient algorithm for convex hulls in two dimensions", *Info. Proc. Letters*, Vol. 9, pp. 216–219.
- Aoyama, Y. and Nakano, J. (1999), *RS/6000 SP: Practical MPI programming*, IBM, Austin, TX.
- Arita, D., Hamada, Y. and Taniguchi, R. (1999), A real-time distributed video image processing system on pc-cluster., *in* 'ACPC', pp. 296–305.
- Arita, D., Hamada, Y., Yonemoto, S. and Taniguchi, R. (2000), Rpv: A programming environment for real-time parallel vision - specification and programming methodology., *in* J. D. P. Rolim, ed., 'IPDPS Workshops', Vol. 1800 of *Lecture Notes in Computer Science*, Springer, pp. 218–225.
- Arita, D. and Taniguchi, R. (2001), Rpv-ii: A stream-based real-time parallel vision system and its application to real-time volume reconstruction., *in* B. Schiele and G. Sagerer, eds, 'ICVS', Vol. 2095 of *Lecture Notes in Computer Science*, Springer, pp. 174–189.
- Arof, H. and Deravi, F. (1998), "Circular neighbourhood and 1-d dft features for texture classification and segmentation", *IEE Proceedings-Vision Image and Signal Processing*, Vol. 145, pp. 167–172.
- Athistos, V., Alon, J. and Sclaroff, S. (2005), Efficient nearest neighbor classification using a cascade of approximate similarity measures, *in* 'CVPR 2005, Int. Conf. on Computer Vision and Pattern Recognition', San Diego, CA, pp. 486–493.
- Athistos, V. and Sclaroff, S. (2001), 3d hand pose estimation by finding appearance-based matches in a large database of training views, *in* 'IEEE Workshop on Cues in Communications', IEEE, pp. 100–106.
- Awcock, G. W. and Thomas, R. (1996), *Applied Image Processing*, McGraw-Hill.

- Balaji, P., Shiam, P., Wyckoff, P. and Panda, D. (2002), High performance user level sockets over gigabit ethernet, *in* ‘Proceedings of the IEEE Int. Conf. on Cluster Computing’, LOCAL?, pp. 179–186.
- Baluja, S. (1999), Making templates rotationally invariant: An application to rotated digit recognition, *in* ‘Advances in Neural Information Processing Systems NIPS ’99’.
- Barbosa, J. G. and Padilha, A. J. (1998), Algorithm-dependant method to determine the optimal number of computers in parallel virtual machines., *in* J. M. L. M. Palma, J. Dongarra and V. Hernández, eds, ‘VECPAR’, Vol. 1573 of *Lecture Notes in Computer Science*, Springer, pp. 508–521.
- Barbosa, J., Tavares, J. and Padilha, A. J. (2001), Parallel image processing system on a cluster of personal computers, *in* J. M. L. M. Palma, J. Dongarra and V. Hernández, eds, ‘VECPAR 2000, 4th Int. Conf.’, Vol. 1981 of *Lecture Notes in Computer Science*, Springer, Porto, Portugal, pp. 439–542.
- Barczak, A. L. C., Messom, C. H. and Johnson, M. J. (2003), Performance characteristics of a cost-effective medium-sized beowulf cluster supercomputer, *in* ‘LNCS 2660’, Springer Verlag, pp. 1050–1059.
- Bookman, C. (2003), *Linux Clustering*, New Riders, Indianapolis.
- Bradski, G. (2000), “The opencv library”, *Dr. Dobb’s Journal*, Vol. 25, pp. 120–126.
- Bradski, G. (2002), Opencv: Examples of use and new applications in stereo, recognition and tracking, *in* ‘Proc. Intern. Conf. on Vision Interface’.
- Burghardt, T., Calic, J. and Thomas, B. (2004), Tracking animals in wildlife videos using face detection, *in* ‘European Workshop on the Integration of Knowledge, Semantics and Digital Media Technology’.
- Burghardt, T., Thomas, B., Barham, P. J. and Calic, J. (2004), Automated visual recognition of individual african penguins, *in* ‘Fifth International Penguin Conference’, Ushuaia, Tierra del Fuego, Argentina.
- Casagrande, N. (2005), Automatic music classification using boosting algorithms and auditory features, Master’s thesis, University of Montreal. Department of Informatic and Operational Research.
- Chen, Y., Zhang, M., Lu, P. and Wang, Y. (2005), Local moment invariant analysis, *in* ‘International Conference on Computer Graphics, Imaging and Visualization (CGIV05)’, pp. 137–140.

- Chong, C., Raveendran, P. and Mukundan, R. (2003), “A comparative analysis of algorithms for fast computation of zernike moments”, *Pattern Recognition*, Vol. 36, pp. 731–742.
- Cristinacce, D. and Cootes, T. (2003), Facial feature detection using adaboost with shape constraints, *in* ‘BMVC2003’, pp. I:231–240.
- Crow, F. C. (1984), “Summed-area tables for texture mapping”, *Computer Graphics*, Vol. 18 (3), pp. 207–212.
- Dadgostar, F. and Barczak, A. L. C. (2005), “A colour hand gesture database for evaluating and improving algorithms on hand gesture and posture recognition”, *Research Letters in the Information and Mathematical Sciences*, Vol. 5, pp. 127–134.
- de Sa, J. P. M. (2001), *Pattern Recognition, Concepts, Methods and Applications*, Springer, Portugal.
- DeRidder, D. (2001), *Adaptative Methods of Image Processing*, PhD thesis, ASCI Delft.
- Dlagnekov, L. (2004), License plate detection using adaboost. Project Report CSE at UC San Diego.
- Dongarra, J. (1989), Performance of various computers using standard linear equations software, Technical Report CS-89-85, University of Tennessee.
- Drummond, C. and Holte, R. C. (2006), “Cost curves: An improved method for visualizing classifier performance”, *Machine Learning*, Vol. 65, pp. 95–130.
- Egan, J. P. (1975), *Signal Detection Theory and ROC Analysis*, Academic Press.
- Flusser, J. (1998), Fast calculation of geometric moments of binary images, *in* M. Gengler, ed., ‘Pattern Recognition and Medical Computer Vision’, ÖCG, Illmitz, pp. 265–274.
- Flusser, J. (2000a), “On the independence of rotation moment invariants”, *Pattern Recognition*, Vol. 33, pp. 1405–1410.
- Flusser, J. (2000b), “Refined moment calculation using image block representation”, *IEEE Trans. on Image Processing*, Vol. 9, pp. 1977–1978.
- Flusser, J. (2002), “On the inverse problem of rotation moment invariants”, *Pattern Recognition*, Vol. 35, pp. 3015–3017.
- Flusser, J. and Suk, T. (1993), “Pattern recognition by affine moment invariants”, *Pattern Recognition*, Vol. 26, pp. 167–174.

- Flusser, J. and Suk, T. (1999), On the calculation of image moments, Research Report 1946, Institute of Information Theory and Automation, Academy of Sciences of the Czech Republic, Prague, Czech Republic.
- Freund, Y. (1998), An introduction to boosting based classification, *in* ‘Proceedings of the AT&T conference on Quantitative Analysis’.
- Freund, Y. and Schapire, R. E. (1996), Experiments with a new boosting algorithm, *in* ‘Proceedings of the 13th International Conference in Machine Learning’, Bari, Italy, pp. 148–156.
- Freund, Y. and Schapire, R. E. (1999), “A short introduction to boosting”, *Journal of Jap. Society for Art. Intell.* , Vol. 14, pp. 771–780.
- Froba, B., Stecher, S. and Kublbeck, C. (2003), Boosting a haar-like feature set for face verification, *in* J. Kittler and M. S. Nixon, eds, ‘LNCS 2688’, AVBPA 2003, Springer-Verlag, pp. 617–624.
- Garcia, C. and Delakis, M. (2004), “Convolutional face finder: A neural architecture for fast and robust face detection”, *IEEE Trans. on Pattern Analysis and Machine Intelligence* , Vol. 26, pp. 1408–1423.
- Gong, S., McKenna, S. J. and Psarrou, A. (2000), *Dynamic Vision: from Images to Face Recognition*, ICP, London.
- Gonzalez, R. C. and Woods, R. E. (2002), *Digital Image Processing*, Prentice Hall.
- Gorgevik, D. and Cakmakov, D. (2004), An efficient three-stage classifier for handwritten digit recognition, *in* ‘17th International Conference on Pattern Recognition (ICPR’04)’, IEEE, pp. 507–510.
- Gropp, W. and Lusk, E. L. (1999), Reproducible measurements of mpi performance characteristics, *in* ‘Proceedings of the 6th European PVM/MPI Users’ Group Meeting on Recent Advances in Parallel Virtual Machine and Message Passing Interface’, Springer-Verlag, London, UK, pp. 11–18.
- Gropp, W., Lusk, E. and Skjellum, A. (1999), *Using MPI*, MIT Press, Cambridge, MA.
- Gropp, W., Lusk, E. and Thakur, R. (1999), *Using MPI-2*, MIT Press, Cambridge, MA.
- Grosz, L. and Barczak, A. L. C. (2000), “Building an inexpensive parallel computer”, *Res. Lett. Inf. Math. Sci* , Vol. 1, pp. 113–118.
- Guo, G. and Zhang, H. (2001), Boosting for fast face recognition, Technical Report MSR-TR-2001-16, Microsoft Research (MSR).

- Hastie, T., Tibshirani, R. and Friedman, J. (2001), *The Elements of Statistical Learning*, Springer-Verlag, New York.
- Hu, M.-K. (1962), “Visual pattern recognition by moment invariants”, *IRE Transactions on Information Theory*, Vol. 8, pp. 179–187.
- Jain, A. K. (1989), *Fundamentals of Digital Image Processing*, Prentice Hall, Upper Saddle River, New Jersey.
- Jones, M. J. and Viola, P. (2003), Fast multi-view face detection, Technical Report TR2003-96, MERL.
- Kazhdan, M., Chazelle, B., Dobkin, D., Funkhouser, T. and Rusinkiewicz, S. (2003), “A reflective symmetry descriptor for 3D models”, *Algorithmica*, Vol. 38.
- Keysers, D. M. (2006), Modeling of Image Variability for Recognition, PhD thesis, RWTH Aachen University,.
- Kolsch, M. and Turk, M. (2004a), Analysis of rotational robustness of hand detection with a viola-jones detector, in ‘ICPR04’, pp. III: 107–110.
- Kolsch, M. and Turk, M. (2004b), Fast 2d hand tracking with flocks of features and multi-cue integration, in ‘RealTimeHCI04’, pp. 158–164.
- Kolsch, M. and Turk, M. (2004c), Robust hand detection, in ‘AFGR04’, pp. 614–619.
- Kotoulas, L. and Andreadis, I. (2006), “Fast computation of chebyshev moments”, *IEEE Trans. on Circ. and Syst. for Video Technology*, Vol. 16, pp. 884–888.
- LeCun, Y., Bottou, L., Bengio, Y. and Haffner, P. (1998), “Gradient-based learning applied to document recognition”, *Proceedings of the IEEE*, Vol. 86, pp. 2278–2324.
- Li, S. Z. (2005), *Face Detection*, Handbook of Face Recognition, Springer, chapter 2, pp. 13–38.
- Li, S. Z. and Zhang, Z. (2004), “Floatboost learning and statistical face detection”, *IEEE Trans. on Pattern Analysis and Machine Intelligence*, Vol. 26, pp. 1112–1123.
- Li, S., Zhu, L., Zhang, Z., Blake, A., Zhang, H. and Shum, H. (2002), Statistical learning of multi-view face detection, in ‘Proc. of the 7th European Conference on Computer Vision’, Copenhagen, Denmark, pp. 67–81.
- Liao, S., Lu, Q. and Lee, K. (1997), Recognition of chinese characters by moment feature extraction, in ‘17th International Conference on Computer Processing of Oriental Languages (ICCPOL 97)’, Hong Kong, pp. 566–571.

- Lienhart, R., Kuranov, A. and Pisarevsky, V. (2003), Empirical analysis of detection cascades of boosted classifiers for rapid object detection, *in* 'DAGM03', Madgeburg, Germany, pp. 297–304.
- Lienhart, R., Liang, L. and Kuranov, A. (2003), A detector tree of boosted classifiers for real-time object detection and tracking, *in* 'ICME2003', IEEE, pp. 277–280.
- Lienhart, R. and Maydt, J. (2002), An extended set of haar-like features for rapid object detection, *in* 'ICIP02', Rochester, NY, pp. I: 900–903.
- Lockton, R. and Fitzgibbon, A. W. (2002), Real-time gesture recognition using deterministic boosting, *in* 'Proceedings, British Machine Vision Conference'.
- Lucas, B. and Kanade, T. (1981), An iterative image registration technique with an application to stereo vision, *in* 'Proc. of 7th International Joint Conference on Artificial Intelligence (IJCAI)', pp. 674–679.
- Luo, D. (1998), *Pattern Recognition and Image Processing*, Horwood Publishing.
- Marr, D. (1982), *Vision*, W.H.Freeman.
- McCane, B. and Novins, K. (2003), On training cascade face detectors, *in* 'Image and Vision Computing New Zealand', Palmerston North, pp. 239–244.
- McCane, B., Novins, K. and Albert, M. (2005), Optimizing cascade classifiers. available at <http://www.cs.otago.ac.nz/staffpriv/mccane/publications.html>.
- Menezes, P., Barreto, J. C. and Dias, J. (2004), Face tracking based on haar-like features and eigenfaces, *in* '5th IFAC Symposium on Intelligent Autonomous Vehicles', Lisbon, Portugal.
- Mita, T., Kaneko, T. and Hori, O. (2005), Joint haar-like features for face detection, *in* '10th IEEE International Conference in Computer Vision (ICCV'05)', IEEE, pp. 1619–1626.
- Mitri, S., Pervlz, K., Surmann, H. and Nchter, A. (2004), "Fast color-independent ball detection for mobile robots", *Mechatronics and Robotics*, pp. 900–905.
- Moga, A. N. (1999), Parallel multiresolution image segmentation with watershed transformation., *in* 'ACPC', pp. 226–235.
- Moga, A. N., Cramariuc, B. and Gabbouj, M. (1998), "Parallel watershed transformation algorithms for image segmentation.", *Parallel Computing*, Vol. 24, pp. 1981–2001.
- Mukundan, R. (2005), Radial tchebichef invariants for pattern recognition, *in* 'Proc. of IEEE Tencon Conference Tencon05', Melbourne, pp. 2098–2103.

- Mukundan, R., Ong, S. H. and Lee, P. A. (2001), “Image analysis by tchebichef moments”, *IEEE Trans. on Image Processing*, Vol. 10, pp. 1357–1364.
- Ng, C. W. and Ranganath, S. (2002), “Real-time gesture recognition system and application.”, *Image Vision Comput.*, Vol. 20, pp. 993–1007.
- Ong, E.-J. and Bowden, R. (2004), A boosted classifier tree for hand shape detection, *in* ‘AFGR04’, IEEE, pp. 889–894.
- Osadchy, R., Miller, M. and LeCun, Y. (2005), Synergistic face detection and pose estimation with energy-based model, *in* ‘Advances in Neural Information Processing Systems (NIPS 2004)’, MIT Press.
- Papageorgiou, C., Oren, M. and Poggio, T. (1998), A general framework for object detection, *in* ‘International Conference on Computer Vision’.
- Phillips, P. J., Moon, H., Rizvi, S. A. and Rauss, P. J. (2000), “The feret evaluation methodology for face recognition algorithms”, *IEEE Trans. Pattern Analysis and Machine Intelligence*, Vol. 22, pp. 1090–1104.
- Postma, E., van den Herik, J. and Hudson, P. (1997), *Brain-like Computing and Intelligent Information Systems*, New York, Springer Verlag, chapter Image Recognition by Brains and Machines, pp. 25–47.
- Reiss, T. H. (1991), “The revised fundamental theorem of moment invariants”, *IEEE Trans. on Pattern Analysis and Machine Intelligence*, Vol. 13, pp. 830–834.
- Reussner, R. (2001), *High Performance Computing in Science and Engineering 2000*, Transactions of the High Performance Computing Center Stuttgart (HLRS), Springer, chapter Recent Advances in SKaMPI, pp. 520–530.
- Ripley, B. D. (1996), *Pattern recognition and neural networks*, Cambridge University Press, Cambridge, New York.
- Rodtook, S. and Makhanov, S. (2005), “Numerical experiments on the accuracy of rotation moments invariants”, *Image and Vision Computing*, pp. 577–586.
- Roerdink, J. B. T. M. and Meijster, A. (2000), “The watershed transform: Definitions, algorithms and parallelization strategies.”, *Fundam. Inform.*, Vol. 41, pp. 187–228.
- Rowley, H. A. (1999), Neural network-based face detection, PhD thesis, Carnegie Mellon University.
- Rowley, H. A., Baluja, S. and Kanade, T. (1998a), Rotation invariant neural network-based face detection, *in* ‘Proceedings of IEEE Conference on Computer Vision and Pattern Recognition’, pp. 38–44.

- Rowley, H., Baluja, S. and Kanade, T. (1998b), “Neural network-based face detection”, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, , Vol. Vol. 20, pp. 23–38.
- Rudin, C., Daubechies, I. and Shapire, R. E. (2004), “The dynamics of adaboost: Cyclic behaviour and convergence of margins”, *Jornal of Machine Learning Research* , Vol. 5, pp. 1557–1595.
- Schneiderman, H. and Kanade, T. (1998), Probabilistic modeling of local appearance and spatial relationships for object recognition, *in* ‘IEEE Conference on Computer Vision and Pattern Recognition (CVPR 1998)’, Santa Barbara, CA, pp. 45–51.
- Schneiderman, H. and Kanade, T. (2000), A statistical method for 3d object detection applied to faces and cars, *in* ‘IEEE Conference on Computer Vision and Pattern Recognition (CVPR 2000)’, pp. 1746–1759.
- Schweitzer, H., Bell, J. W. and Wu, F. (2002), Very fast template matching, *in* A. H. et al., ed., ‘LNCS 2353’, ECCV 2002, Springer-Verlag, pp. 358–372.
- Simard, P. Y., LeCun, Y. A. and Denker, J. S. (1992), Advances in neural information processing systems, *in* ‘NIPS (Advances in Neural Information Processing Systems) 1992’, Vol. 5, Denver, Colorado, pp. 50–58.
- Sloan, J. (2004), *Linux Clusters*, O’Reilly, Sebastopol, CA.
- Spiliotis, I. M. and Mertzios, B. G. (1998), “Real-time computation of two-dimensional moments on binary images using image block representation”, *IEEE Trans. on Image Processing* , Vol. 7, pp. 1609–1615.
- Stenger, B. D. R. (2004), Mode-Based Hand Tracking Using a Hierarchical Bayesian Filter, PhD thesis, University of Cambridge.
- Sterling, T. L., Savarese, D., Becker, D. J., Dorband, J. E., Ranawake, U. A. and Packer, C. V. (1995a), Beowulf: a parallel workstation for scientific computation, *in* ‘Proceedings of the International Conference on Parallel Processing (ICPP)’, Urbana-Chanpain, Illinois, pp. 11–14.
- Sterling, T., Salmon, J., Becker, D. and Savarese, D. (1999), *How to build a Beowulf: a guide to the implementation and application of PC clusters*, MIT Press, Cambridge, MA.
- Sterling, T., Savarese, D., Becker, D. J., Dorband, J. E., Ranawake, U. A. and Packer, C. V. (1995b), BEOWULF: A parallel workstation for scientific computation, *in* ‘Proceedings of the 24th International Conference on Parallel Processing’, Oconomowoc, WI, pp. I:11–14.

- Suen, C. Y. and Tan, J. (2005), “Analysis of errors of handwritten digits made by a multitude of classifiers”, *Pattern Recognition Letters*, Vol. 26, pp. 369–379.
- Suk, T. and Flusser, J. (2003), “Combined blur and affine moment invariants and their use in pattern recognition”, *Pattern Recognition*, pp. 2895–2907.
- Sun, Y., Todorovic, S. and Li, J. (2006), “Reducing the overfitting of adaboost by controlling its data distribution skewness”, *International Journal of Pattern Recognition and Artificial Intelligence*, Vol. 20, pp. 1093–1116.
- Sung, K. and Poggio, T. (1998), “Example-based learning for view-based face detection”, *IEEE Patt. Anal. Mach. Intell.*, Vol. 20.
- Teh, C. and Chin, R. (1988), “On image analysis by the methods of moments”, *PAMI*, Vol. 10, pp. 496–513.
- Terrillon, J.-C., McReynolds, D., Sadek, M., Sheng, Y. and Akamatsu, S. (2000), Invariant neural-network based face detection with orthogonal fourier-mellin moments., *in* ‘International Conference on Pattern Recognition ICPR2000’, pp. 993–1000.
- Terrillon, J.-C., Pilpre, A., Niwa, Y. and Yamamoto, K. (2004), *Druide* : A real-time system for robust multiple face detection, tracking and hand posture recognition in color video sequences., *in* ‘ICPR 2004 (3)’, pp. 302–305.
- Terrillon, J., David, M. and Akamatsu, S. (1998), Automatic detection of human faces in natural scene images by use of a skin color model and of invariant moments, *in* ‘AFGR98’, pp. 112–117.
- Thacker, N., Clark, A., Barron, J., Beveridge, R., Clark, C., Courtney, P., Crum, W. and Ramesh, V. (2005), Performance characterisation in computer vision: A guide to best practices, Technical Report 2005-009, Medical School, University of Manchester, Manchester, UK.
- Top 500 computers* (n.d.).
URL: <http://www.top500.org>
- Torres-Mendez, L., Ruiz-Suarez, J. C., Sucar, L. E. and Gomez, G. (2000), “Translation, rotation, and scale-invariant object recognition”, *IEEE Transactions on Systems, Man and Cybernetics*, Vol. 30, pp. 125–130.
- Trier, O., Jain, A. and Taxt, T. (1996), “Feature-extraction methods for character-recognition: A survey”, *Pattern Recognition*, Vol. 29, pp. 641–662.
- Viola, P. and Jones, M. (2001a), Rapid object detection using a boosted cascade of simple features, *in* ‘CVPR01’, IEEE, Kauai, HI, pp. I:511–518.

- Viola, P. and Jones, M. (2001*b*), Robust Real-Time face detection, *in* ‘Proceedings of the Eighth International Conference On Computer Vision (ICCV-01)’, IEEE Computer Society, Los Alamitos, CA, pp. 747–747.
- Viola, P. and Jones, M. (2004), “Robust real-time face detection”, *International Journal of Computer Vision*, Vol. 57, pp. 137–154.
- Viola, P., Jones, M. and Snow, D. (2005), “Detecting pedestrians using patterns of motion and appearance”, *IJCV*, Vol. 63, pp. 153–161.
- Wilkinson, B. and Allen, M. (1999), *Parallel Programming*, Prentice Hall, New Jersey.
- Wong, W.-H., Siu, W.-C. and Lam, K.-M. (1995), “Generation of moment invariants and their uses for character recognition”, *Pattern Recognition Letters*, pp. 115–123.
- Wood, J. (1996), “Invariant pattern recognition: a review”, *Pattern Recognition*, Vol. 29, pp. 1–17.
- Wu, J., Rehg, J. M. and Mullin, M. D. (2003), Learning a rare event detection cascade by direct feature selection, *in* ‘NIPS (Advances in Neural Information Processing Systems) 2003’, Vancouver, Canada.
- Yang, L. and Albrechtsen, F. (1996), “Fast and exact computation of cartesian geometric moments using discrete greens theorem”, *Pattern Recognition*, Vol. 29, pp. 1061–1073.
- Yang, M.-H., Kriegman, D. J. and Ahuja, N. (2002), “Detecting faces in images: A survey”, *IEEE Trans. on PAMI*, Vol. 24, pp. 34–58.
- Zaki, O., E., L., Gropp, L. and Swider, D. (1999), “Toward scalable performance visualization with jumpshot.”, *High Performance Computing Applications*, Vol. 13, pp. 277–288.
- Zhu, J., Rosset, S., Zou, H. and Hastie, T. (2006), Multi-class adaboost, Technical Report 430, Department of Statistics, University of Michigan.
- Zhu, Z., Morimoto, T., Adachi, H. and Kiriya, O. (2004), Multi-view face detection and recognition using haar-like features, *in* ‘Third Hiroshima International Workshop on NTIP’, IEEE, pp. 109–111.

Glossary

Accuracy of a classifier Indicates the hit rate and false detection rate of a classifier 11

AdaBoost Training algorithm that produces classifiers based on weak classifiers. 18

Bandwidth Average data transmission rate (in bits/second or in Bytes/second). 30

Beowulf Clusters Cluster of computers built from off-the-shelf components. Usually the communication between the nodes is done via a dedicated network. 29

Binary Classifiers A classifier that distinguishes all the elements of a set into two classes. 10

CDMI Concentric Discs Moment Invariants refer to both the method and the feature values. The moment invariant features are extracted from discs of the image using a number of SATs. 115

Cascade A classifier that is formed by several layers or stages, each layer being a classifier itself. Each layer only analyses the sub-windows that passed all the previous layers. 20

Central Moments Simple form of geometric moments that are translation invariant. 104

Cluster Topology Network topology that links the nodes that composes a Beowulf cluster. 29

Concurrent Cascades Classifiers based on cascades that run on the same processor and share one or more SATs. 149

Contrast Stretching Operation that spread the pixel values of an image or a sub-window over the range limited by the size of the representation. Similar to a histogram equalisation. 26

Convex Hull The smallest set of points that contains a set of points P and is a sub-set of P. 42

Data Decomposition The division of the work among the nodes of a cluster is done based on the data, so that usually the nodes repeat the same functions with different data. 28

FERET Facial Image Database produced at the National Institute of Standards and Technology (NIST). 58

False Detection Rate % of sets incorrectly classified as positive elements of the test set. 11

Function Decomposition The division of the work among the nodes of a cluster is done based on function, so the nodes usually share the same data. 28

Haar-like features Feature set based on functions similar to the Haar basis functions. 15

Helix A 128 processors Beowulf cluster built in 2003 at Massey University. 49

Hit Rate % of correctly classified elements of the test set. 11

Integral Image A data structure that allows fast computation of sum of pixels at different scales and positions. 16

Invariant Features Features that present the same values when faced with geometric transformations. 8

Kernel The basic sub-window for multiresolution analysis, kernel is also the window size of the samples with which the classifiers were trained. 24

Latency The time delay to start to transmit a message (in seconds or msec). 30

Layer See stage.20

Linpack An accepted standard benchmark to compare the performance among parallel machines. 49

MNIST MNIST or Mini-NIST refer to a sub-set of the image database produced by NIST. The MNIST contains only hand-written digits and is composed of 60000 training images and 10000 test images. 110

Message Passing A programming paradigm that facilitates the communication between the nodes of a cluster. The two most common libraries that implement such a paradigm are MPI and PVM. 32

Mobile Cluster A mini Beowulf cluster based on mini-ITX boards and fed by batteries. 146

- Moment Invariants order** Indicates the exponent used to each coordinate when computing 2D moments (p for the x and q for the y axis). 103
- Moment Invariants** Features extracted from 2D images based on the moments given by the value and coordinates of each pixel. 103
- Negative examples, negative set, negative samples** A set of values obtained by feature extraction from images that are not to be detected, such as sub-windows that are part of the background or that represent other objects. 18
- Non-invariant Features** Features that change values when faced with geometric transformations. 8
- Normalised Central Moments** Simple form of geometric moments that are translation and scaling invariant. 104
- Object Detection** Locate generic classes of objects in the image (such as faces). 10
- Object Recognition** Classify specific objects in the image, such as a face that belongs to one individual, or a certain printed character etc. 10
- Object Tracking** dynamically locates objects by determining its position in each frame. 10
- OpenCV** An open-source computer vision library developed at Intel (see Bradski (2000)). 12
- PEF** Pair of Equivalent Features, a pair of Haar-like features that makes an approximation to an arbitrary angle. 84
- Partial Occlusion** The object to be detected or recognised is partially covered by another object. 57
- Performance of a classifier** Indicates the speed of a classifier. Can be measure in seconds per image or in frames per second. 11
- Positive examples, positive set, positive samples** A set of values obtained by feature extraction from images that are to be detected or recognised. 18
- ROC curves** Receiver operating characteristic curves draws a function of the Hit rate x False Detection rate. 11
- Rotation** Geometric rotation of the kernel within the constraints of a digital image. 25
- SAT** Summed-area Tables, also known as Integral Images. A data structure that allows fast computation of sum of pixels at different scales and positions. 16

Scaling factor The multiplicative factor that defines the sub-window size in relation to the kernel size. 24

Scaling The resolution of the kernel is the scale of the sub-window that is being analysed. 24

Sisters A 16 processor Beowulf cluster built in 2000 for the Institute of Information and Mathematical Sciences at Massey University. 49

Speedup Indicates the gain in performance when a job is shared across two or more nodes (in speed or runtime). 31

Stage A set of weak classifier that together composes a strong classifier. Several stages in serial forms a cascade of classifiers. Also called “layer”. 39

Tilted Haar-like features Haar-like features computed at 45° . 16

Translation factor The displacement between sub-windows during the detection process, measure in pixels. 24

Translation The displacement of the kernel over the image. 24

Viola-Jones Method A feature-based object detection method that uses Haar-like features and AdaBoost. 15

Weak Classifiers A binary classifier based in simple rules that performs slightly better than 50%. 18