


Detection of Black-Backed Jackal in Still Images

by

Sneha P Pathare

*Thesis presented in partial fulfilment of the requirements
for the degree Master of Science in Electronic Engineering
at the University of Stellenbosch*

The crest of the University of Stellenbosch is centered behind the text. It features a shield with various symbols, topped with a crown and surrounded by a decorative wreath. Below the shield is a motto scroll.

Department of Electrical and Electronic Engineering,
University of Stellenbosch,
Private Bag X1, Matieland 7602, South Africa.

Supervisor: Prof J.A. du Preez

March 2015

Declaration

By submitting this thesis/dissertation electronically, I declare that the entirety of the work contained therein is my own, original work, that I am the sole author thereof (save to the extent explicitly otherwise stated), that reproduction and publication thereof by Stellenbosch University will not infringe any third party rights and that I have not previously in its entirety or in part submitted it for obtaining any qualification.

January 2015

Copyright © 2015 Stellenbosch University
All rights reserved

Abstract

In South Africa, black-back jackal (BBJ) predation of sheep causes heavy losses to sheep farmers. Different control measures such as shooting, gin-traps and poisoning have been used to control the jackal population; however, these techniques also kill many harmless animals, as they fail to differentiate between BBJ and harmless animals. In this project, a system is implemented to detect black-backed jackal faces in images. The system was implemented using the Viola-Jones object detection algorithm. This algorithm was originally developed to detect human faces, but can also be used to detect a variety of other objects. The three important key features of the Viola-Jones algorithm are the representation of an image as a so-called "integral image", the use of the Adaboost boosting algorithm for feature selection, and the use of a cascade of classifiers to reduce false alarms.

In this project, Python code has been developed to extract the Haar-features from BBJ images by acting as a classifier to distinguish between a BBJ and the background. Furthermore, the feature selection is done using the Asymboost instead of the Adaboost algorithm so as to achieve a high detection rate and low false positive rate. A cascade of strong classifiers is trained using a cascade learning algorithm. The inclusion of a special fifth feature Haar feature, adapted to the relative spacing of the jackal's eyes, improves accuracy further. The final system detects 78% of the jackal faces, while only 0.006% of other image frames are wrongly identified as faces.

Opsomming

Swartrugjakkalse veroorsaak swaar veë-verliese in Suid Afrika. Teenmaatreels soos jag, slagysters en vergiftiging word algemeen gebruik, maar is nie selektief genoeg nie en dood dus ook vele nie-teiken spesies. In hierdie projek is 'n stelsel ontwikkel om swartrugjakkals gesigte te vind op statiese beelde. Die Viola-Jones deteksie algoritme, aanvanklik ontwikkel vir die deteksie van mens-gesigte, is hiervoor gebruik. Drie sleutel-aspekte van hierdie algoritme is die voorstelling van 'n beeld deur middel van 'n sogenaamde integraalbeeld, die gebruik van die "Adaboost" algoritme om gepaste kenmerke te selekteer, en die gebruik van 'n kaskade van klassifiseerders om vals-alarm tempos te verlaag.

In hierdie projek is Python kode ontwikkel om die nuttigste "Haar"-kenmerke vir die deteksie van dié jakkalse te onttrek. Eksperimente is gedoen om die nuttigheid van die "Asymboost" algoritme met die van die "Adaboost" algoritme te kontrasteer. 'n Kaskade van klassifiseerders is vir beide van hierdie tegnieke afgerig en vergelyk. Die resultate toon dat die kenmerke wat die "Asymboost" algoritme oplewer, tot laer vals-alarm tempos lei. Die byvoeging van 'n spesiale vyfde tipe Haar-kenmerk, wat aangepas is by die relatiewe spasieëring van die jakkals se oë, verhoog die akkuraatheid verder. Die uiteindelige stelsel vind 78% van die gesigte terwyl slegs 0.006% ander beeld-raampies verkeerdelik as gesigte geklassifiseer word.

Acknowledgements

I would like to use this opportunity to express my gratitude to these wonderful people mentioned below who directly and indirectly have supported me throughout this project.

- Foremost, I would like to express my sincere gratitude to my supervisor, Prof J.A. du Preez, for his continuous support, his patience, motivation, enthusiasm and immense knowledge. I am thankful for his friendly advice and guidance at every point during the research and writing of this thesis. One simply could not wish for a better or friendlier supervisor.
- A very special word of thanks to my parents and parents-in-law for their endless love, and the moral support and understanding they have provided me throughout my entire life.
- My sincere thanks also go to the DSP lab members for making coffee and keeping the network running. A special thanks to my dearest friend, Hlonie Mohasi, for her moral support and encouragement. I would also like to thank my friends Lerato Oa Lerato, Harry Mafukidze and Jedri Visser for their help and friendly behaviour.
- I would also like to thank Dr Riaan Wolhuter, Dr Johann Strauss and, again, my supervisor Prof J.A. du Preez, for helping me to collect the database of jackal by providing pictures and videos of jackal.
- I also recognise that this research would not have been possible without the financial assistance of OSP-CIT theme funds and I express my gratitude to them.
- Last but not least, I must acknowledge my beloved husband Pankaj, without whose love, support, encouragement and editing assistance, I would not have finished this thesis.

Contents

Declaration	v
Abstract	v
Opsomming	v
Acknowledgements	v
Contents	v
List of Figures	viii
List of Tables	x
List of Symbols	xi
Nomenclature	xii
1 Introduction	1
1.1 Background	1
1.2 Motivation	3
1.2.1 Black-backed Jackal:	3
1.2.2 Black-backed Jackal as a Problem Animal:	4
1.2.3 Control Measures:	5
1.3 Literature Synopsis	7
1.3.1 Animal Detection by Human Eyes	7
1.3.2 Animal Detection Based on the Human Face Detection Approach	8
1.3.3 Animal Detection Based on the Threshold Segmentation Method	8
1.4 Objectives	8
1.5 Contributions	9
1.6 Overview	9
2 Literature Review	11
2.1 Introduction	11
2.2 Animal Detection in Images and Videos	12
2.2.1 Animal Detection by Human eyes	12

2.2.2	Animal Detection Based on Power Spectrum	13
2.2.3	Animal Detection Based on Threshold Segmentation	13
2.2.4	Animal Detection Based on a Human Face Detection Approach	14
2.3	Human Face Detection	14
2.3.1	Challenges Faced by Human Face Detection	15
2.3.2	Face Detection Algorithm	16
2.3.3	Applications of Face Detection	23
2.4	Existing Face Detection Method	23
2.5	Summary	24
3	The Viola-Jones Face Detector	26
3.1	Introduction	26
3.2	Weak Classifier: Haar Features	27
3.2.1	Upright Haar Feature	27
3.2.2	Parity of Haar Feature	30
3.2.3	Haar Feature Threshold	31
3.2.4	Discussion of the Haar Feature	32
3.3	Strong Classifier	32
3.3.1	AdaBoost	33
3.3.2	AsymBoost	35
3.4	Cascading Strong Classifiers	35
3.5	Variance Normalisation	37
3.6	Scanning of a Detector Window	38
3.7	Integration of Multiple Detections	38
3.8	Summary	39
4	Human Face Detection	40
4.1	Introduction	40
4.2	Preparation of Database	41
4.2.1	Available Human Face Databases	41
4.2.2	Negative Database	43
4.3	Implementation of the Human Face Detection System	43
4.3.1	Pre-Processing of the Image	44
4.3.2	Implementation	44
4.4	Tools Used in Project	45
4.4.1	Hardware	45
4.4.2	Software	45
4.5	Experiments	46
4.5.1	Experiment 1:	46
4.5.2	Experiment 2:	47
4.6	Final Face Detector	49

4.7	Results and Discussion	49
4.8	Summary	54
5	Jackal Database Preparation	55
5.1	Introduction	55
5.2	Collection of BBJ Images	55
5.2.1	Downloading from the Internet	55
5.2.2	Taking Photographs with a Camera	56
5.2.3	Extracting from Videos	57
5.3	Preparation of Database	57
5.3.1	Generating Positive Samples	58
5.3.2	Generating Negative Samples	59
5.4	Image Processing	59
5.4.1	Greyscale Conversion	60
5.4.2	Image Cropping and Resizing	60
5.4.3	Variance Normalisation	61
5.5	Selection of BBJ Database	61
5.6	Summary	62
6	Jackal Face Detection	63
6.1	Introduction	63
6.2	Implementation of Jackal Detection System	64
6.2.1	Pre-Processing of Images	64
6.2.2	Haar Feature File	65
6.2.3	Implementation	65
6.3	Experiments	65
6.3.1	Experiment 1	66
6.3.2	Experiment 2	67
6.4	Result and Discussion	69
6.5	Summary	71
7	Conclusion and Future Work	73
7.1	Conclusion	73
7.2	Future Work	74
	Bibliography	75
A	OpenCV Haartraining	83
A.1	Preparation of Data	83
A.2	Training	85
A.3	Testing	86

List of Figures

1.1	The above graph shows the number of sheep available.	2
1.2	Black-Backed Jackal.	4
2.1	Example of pose variation.	15
2.2	Example of different illumination condition.	16
2.3	Example of different facial expression.	16
2.4	Example of face occlusion.	16
2.5	A common face template used in knowledge based methods [1].	17
2.6	A 14*16 pixel ratio template for face localization based on the Sinha method	19
2.7	Simple straight line as simple classifier.	20
2.8	Optimal hyperplane as SVM classifier.	20
2.9	Eigenfaces	21
2.10	The ensemble of classifiers.	22
3.1	Four types of Haar feature.	28
3.3	The value of the integral image at a point (x, y)	29
3.4	Original image and its integral image.	30
3.5	The sum within rectangle D can be computed as: $D = ii_4 + ii_1 - (ii_2 + ii_3)$	30
3.6	Direct Haar feature and its inverted counterpart.	31
3.7	Strong classifier loaded consisting of several weak classifiers.	33
3.8	Classification using AdaBoost algorithm [2].	33
3.9	The Cascade of classifiers.	36
4.1	Example image for human face detection.	40
4.2	Example images from FERET database.	41
4.3	Example images from BioID database.	42
4.4	Example images from CMU database.	42
4.5	Face and Nonface example images from the MIT-database.	43
4.6	Background images for human face detection system.	43
4.7	ROC curve for four stage Adaboost and Asymboost systems.	47
4.8	ROC curve for Adaboost, Asymboost and OpenCV systems	48
4.9	An example of images from the BioID test-set.	51
4.10	A - An example of images from the CMU test-set.	52
4.11	B - An example of images from the CMU test-set.	53

5.1	Images collected from the internet.	56
5.2	Images taken with a camera.	56
5.3	Examples of images extracted from videos.	57
5.4	Original image and image rotated by 6°.	58
5.5	Original image and flipped image.	58
5.6	Background images for BBJ based training system.	59
5.7	Original image and its greyscale version.	60
5.8	Cropped and resized image of BBJ.	60
5.9	Original image and variance normalised image.	61
5.10	ROC curve for two different training datasets of BBJ	62
6.1	Examples of images of BBJ from test data set.	64
6.2	ROC curve for system using a basic set of Haar features	67
6.3	Idea elaborating the spacing between human and jackal eyes.	67
6.4	Upright Haar feature.	67
6.5	ROC curve for four and five types of Haar features.	68
6.6	An example images from the BBJ test-set.	69

List of Tables

1.1	The declining economic importance of agriculture [3].	2
2.1	Use of boosting algorithms for face detection.	22
3.1	AdaBoost algorithm [4].	34
3.2	Cascade learning algorithm [5].	37
4.1	Relationship of number of Haar features to the particular size of a window. . . .	45
6.1	This table illustrates the description given in Figure 6.6.	70

List of Symbols

I_n	Intensity of pixel
B	Box sum
w	width of sub-window
h	height of sub-window
I	Original image
II	Integral image
S	Cumulative row sum
h_j	Weak classifier
p_j	Parity of Haar-feature
f_j	Haar-feature (with set size, shape and location)
Θ_j	Threshold of Haar-feature
T_p	Total sum of positive example weights
T_n	Total sum of negative example weights
S_p	Sum of positive weights below the threshold
S_n	Sum of negative weights below the threshold
e	Error respected to the each feature value
T	Total number of rounds in boosting

W_t	Weight of example in t round
y_i	Label of example i
k	Level of asymmetry
m	Number of negative examples
l	Number of positive examples
ε_t	Weighted error of Haar-feature in t round
α_t	Weight of weak classifier in t round
P	Set of positive examples
N	Initial set of negative examples
D	Database of negative examples
F_p	False positive rate of cascade classifier
F_i	False positive rate of i^{th} classifier
D_t	Detection rate of the cascaded classifier
D_i	Detection rate of the i^{th} classifier
σ^2	Variance
n_p	Total number of pixels in detector window
μ	Mean of pixel values
x_p	Pixel value
σ	Standard deviation
s	Current scale of detector window
Δ	Number of pixels

Nomenclature

AVC	Animal-Vehicle Collision
BBJ	Black-backed Jackal
BBS	Basic Background Subtraction
CVAAS	Camel Vehicle Accident Avoidance System
GDP	Gross Domestic Product
GPS	Global Positioning System
ha	Hectares
IE	Inference Engine
IUCN	International Union For Conservation Of Nature
LIDAR	Light Detection And Ranging
LOTS	Lehigh Omnidirectional Tracking System
MMS	Mobile Monitoring System
NEMBA	National Environmental Management Biodiversity Act
NN	Neural Network
OOB	Out Of Bag
PCA	Principal Component Analysis
rf	Random Forest

RFID Radio-Frequency Identification

ROC Receiver Operating Characteristic

RT Random Trees

SGM Single Gaussian Model

SVM Support Vector Machine

V-J Viola-Jones

Chapter 1

Introduction

1.1 Background

Covering 69% [6] of all land surface suitable for grazing, livestock farming (cattle, poultry, sheep, goats, horses and pigs) is one of the most important and largest agricultural sectors in South Africa. Cattle are concentrated in the eastern and wetter regions of the country, as well as in the North West Province and the Northern Cape. Sheep farming is largely practised in the drier western and central areas of the country. With various products such as wool, skin, milk and meat, sheep make a valuable contribution to the South African agricultural sector. However, during the past century, the contribution of sheep products to the total gross agricultural output has steadily dropped from 15.2% in 1948 to 3.7% in 2011 (Table 1.1). This has resulted in a decline of their agricultural contribution to the GDP from 21% to 2% [3]. The number of sheep per person has also declined from 5.1 in 1911 to 0.4 in 2011 (Fig. 1.1). The number of sheep available in South Africa in 2011 was lower than that of previous years. One major contributor to this is sheep losses caused by predation and stock theft. Of these, predation is the more severe one [7, 3].

Sheep farmers are always affected to a greater degree by predators, and the animals responsible for this are termed problem animals or damage-causing animals. Such animals cause huge losses to stock animals or wild species, may cause excessive damage to cultivated crops, trees, natural fauna and other property, and even present a threat to human life (National Environmental Management Biodiversity Act, Act 10 of 2004 (NEMBA)). In South Africa, livestock farmers, especially sheep farmers, have a serious problem with predators. Due to predation:

Table 1.1: The declining economic importance of agriculture [3].

Year	% National population rural	Agriculture as % GDP	Number of commercial farms	Average size of commercial farms (ha)	Farm employees and domestic workers on farms	Wool, lamb and mutton as % of gross agricultural output
1911	75.3	21.0				
1946	63.7	13.0	112,453	837		15.2(1948)
1960	53.3	12.3	105,859	867	907,705	17.0
1970	52.2	8.2	91,154	979	1,299,850	12.0
1980	51.6	7.1	69,372	1,252	1,235,200	6.60
1990	48.0	4.6	62,084	1,335	1,184,700	7.80
1994	46.0	4.6	57,980(1993)	1,427(1993)	921,700	4.7(1993)
2000	43.1	3.30	45,818(2002)		977,610*	3.70
2007	39.8	3.40	39,966		773,900	4.0
2011	38.0	2.40				3.70

Sources: South African Statistics 1964, 1978, 1982; World Development Indicators, Abstract of Agricultural Statistics 2012 (Department of Agriculture, Forestry and Fisheries). *Estimated from average trend between 1990 and 2002.

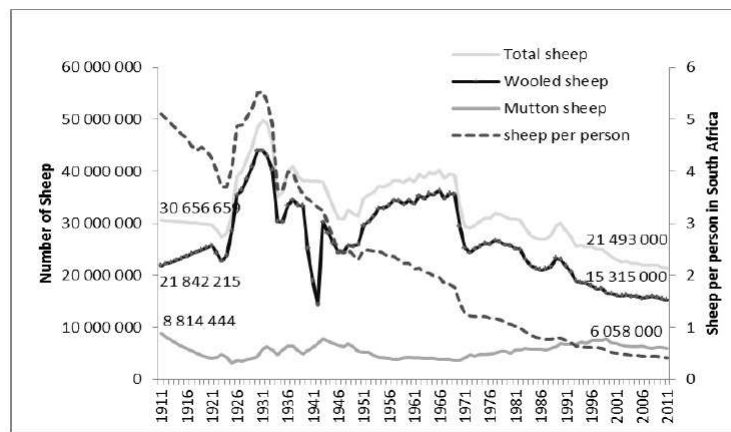


Figure 1.1: The above graph shows that the number of sheep available per person and the total number of sheep in South Africa shows a continuous downward trend from 1911 to 2011 [3].

1. In the Free State province in 2009, out of total sheep losses, 44% was due to predation, 39% was due to disease, and 10% was the result of theft. [8].
2. In KwaZulu-Natal province, losses due to predation were between 0.05% [9] and 29% [10] of the sheep population.

3. In the Northern Cape province, small stock farmers suffered 14% total production loss due to predation [11].
4. The contribution of sheep to the total gross agricultural output has declined drastically [3].

1.2 Motivation

In South Africa the conflict between sheep farmers and predators is ongoing. Sheep farmers suffer heavy and unaffordable livestock losses due to predation [12]. The predators responsible for these are the black-backed jackal (BBJ), caracal and, to a very small extent, leopard. Of these, the jackal is the main culprit, killing 6500 sheep and goats each day across the country [11]. Of the total losses of sheep and goats experienced, around 55% [11] are killed by the jackal alone. Many farmers are using techniques such as gin traps, shooting and trapping to kill jackals, but in the process they also kill many non-targeted animals such as the Cape fox, bat-eared fox, leopard, etc. Heavy losses experienced by sheep farmers and the small stock industry were the main motivation for this study. Additionally, preservation of the harmless creatures that are killed while using control techniques against black-backed jackals also motivate this study.

1.2.1 Black-backed Jackal:

The black-backed jackal (BBJ) (*Canis Mesomeals*, Mammalia:Canidae), also known as the silver-backed jackal and rooijakkals (red jackal in Afrikaans), is a slender, long-legged jackal with large ears and a pointed muzzle, reddish flanks and limbs, and a dark saddle of long (40-60 mm) black and white hair, giving it a silver appearance and hence the name. They are mostly found in the Southern and Eastern part of the African continent, specifically in open savannah and semi-arid zones. It has a bushy tail with a distinctive black sub-caudal marking and a small vertical stripe behind the shoulder and in front of the saddle stripe. BBJ occur in a wide range of habitats, from arid coastal desert to Montane grassland, open savannah, arid savannah, scrub land and farmlands and have a renal adaptation which allows them to survive under arid conditions. BBJ prefer open habitats such as grasslands and overgrazed areas where smaller prey (e.g. rodents) are more vulnerable and easier to prey on [13].



Figure 1.2: Black-Backed Jackal.

1.2.2 Black-backed Jackal as a Problem Animal:

Black-backed jackals eat whatever prey is available, and in farming areas their main target is sheep. They are considered vermin by sheep farmers. The sheep farmers are facing large and unaffordable losses because of the BBJ and hence it has been declared a problem animal in South Africa. Problem animals are defined as ‘animals that cause losses to livestock and wildlife species, damage to crops or property, present a threat to human life or are present in such numbers that agricultural livelihoods are materially depleted’ (National Environmental Management Biodiversity Act, Act 10 of 2004 (NEMBA)). The BBJ is least threatened and has the lowest vulnerability rates of any African predators, and hence is listed as of least concern in the IUCN red list of threatened species [14, 15]. Previously jackals were important, since they preyed on smaller prey on farms (rodents) and maintained the ecological balance, but ‘with the coming of civilization’ they lost their usefulness [16].

Many studies show that BBJ cross reserve boundaries into the neighbouring farmland, where they prey on livestock and this is supported by the studies which have found sheep remains in BBJ stomachs [17]. Predation by jackals on sheep resulted mainly because of a decline in other food available to them. Jackal predation tends to be worst in drought years, when jackals have less to eat and weakened sheep are an easy target [16]. Their reproductive cycle is closely related to that of sheep and hence lambs are available at peak jackal demand. Veld burning

is one of the reasons for the predation, as it kills many small mammals and causes jackals to focus their attention on sheep rearing areas for food. On a specific farm, the loss recorded ranges from 3.9% to 18%, which entails a high economic loss [14]. The BBJ show a number of behavioural features that distinguish them from other predators. Jackals are active during both night and day and can eat a wide range of foods, can adapt to a variety of changing habitats and have a considerably wider range of hunting operations than most other predators. Jackals are endowed with sensitive senses of smell, taste and hearing. Their sensitive sense of taste helps them to detect poisoned food and hence they can easily avoid it [16]. In spite of the use of numerous control measures, BBJ are abundant in Southern Africa, mostly due to their ability to adapt to a wide range of conditions and changing environment, which allows them to expand their ranges and sustain high and stable population sizes.

1.2.3 Control Measures:

‘For 300 years the South African government has helped in trying to eradicate the jackal and failed miserably’ [18]. A number of control measures were used to control the predator (BBJ) population, which has included both lethal and non-lethal control methods.

1. Non-lethal control methods:

Non-lethal control methods mainly include jackal proof fences, guard animals and the King collar.

- Jackal proof fences:

Jackal-proof fences are considered a first line of defence against predators. Woven wire fences and electric fences are examples of jackal-proof fences that restrict the entry of predators into a sheep-rearing area. Although jackal-proof fences work best to protect sheep, there are some limitations to consider. The fencing material is quite expensive and needs regular maintenance. Therefore predator-proof fences are practically useful only for small camps and not for a large lambing camp. Fences also restrict the entry of harmless creatures, causing an ecological imbalance. Also, jackals can climb the fences and can burrow under them; in such cases fencing can be practically useless [16].

- Guard animals:

The guard animal is one of the most effective solutions to reduce predation. Guard animals include the use of guard dogs such as Anatolian dogs, llamas and donkeys. Among these, guard dogs are most popular. Guard dogs are herd-oriented and bond well with the herd. They stay near the sheep and aggressively repel predators [19]. One of the limitations of using guardian animals is that they can protect fewer than 200 sheep in a small open camp. Again, if they get sick or find the company of their own kind, they leave the herd unprotected and thus do not constitute a practical solution for the farmers having more than a thousand sheep.

- King collar:

The King collar is a simple and cheap device to protect sheep. The collar is fitted on a sheep's neck, covering the cheek and underside of the neck. This device is useful to protect sheep, since most predators attack the sheep's throat. However, the device has not proved to be very effective with jackals, as they learn to attack the sheep from the back, causing more trauma to the sheep [18].

2. Lethal control methods:

Lethal methods of predator control are necessary when non-lethal methods fail to control predators. Shooting, trapping, and poisoning are some of the effective lethal control methods employed.

- Shooting:

Hunting is one of the effective ways of lethal control if it is done by professional hunters. However, hiring professional hunters is expensive. Mostly, the hunting is done by the farmers themselves and many non-targeted animals are killed in this process.

- Trapping:

Leg-hold traps or gin traps are the most popular traps used to kill predators. The gin trap is made up of a spring and trigger plate. When the animal steps on the trigger plate the trap closes around the foot, eventually resulting in the death of animal. The major disadvantage of using gin traps is that it kills many non-targeted animals,

since such a device fails to distinguish between jackals and harmless creatures. This leads to the death of many non-targeted animals such as the Cape fox, Bat-eared fox and antelope [20].

The lethal methods of predator control have received vehement opposition from certain sectors of society, such as environmentalist and animal-welfare associations, whereas non-lethal methods are only effective for a limited period of time. With both lethal and non-lethal control methods, animals that are repeatedly in contact with the obstacle learn to avoid them. Considering the losses suffered by sheep farmers, it is seen as justified to kill jackals but it is not legal to kill any other harmless animals. The study of the jackal's habitat and knowing the extent of jackal predation over sheep, as well as a study of the jackal, is a primary need. The specific aim of this study is to define a way in which to safeguard the other species that are killed while hunting and trapping jackal, since these other species play an important role in maintaining ecological balance.

1.3 Literature Synopsis

Animal-based research always play a significant role in any study of the behaviour of targeted animals and the role they play as predator. Many algorithms have been developed for better understanding and management of animals. These algorithms mainly include three branches, namely the detection, tracking, and identification of animals. Animal detection is the very first and most important step in animal-based research. A system that can detect and monitor a targeted animal has various applications and is also helpful in preventing the entry of dangerous and problem animals into residential areas. Animal detection is one of the major topics of research in image processing. Different methods have been developed to detect animals by using images and videos.

1.3.1 Animal Detection by Human Eyes

Early research mainly focused on how fast the animal detection is performed by human eyes. From the point of view of speed and accuracy, animal detection by human eyes is considered the most reliable method. Human eyes can detect whether a scene contains an animal or not within

150 ms [21]. Even though human eyes can achieve highly reliable and satisfactory results, they cannot be used for continuous or large-scale animal detection tasks. Human eyes easily tire and can not work for 24 hours a day, which affects their effectiveness. This motivates researchers to apply computer vision algorithms for animal detection purposes.

1.3.2 Animal Detection Based on the Human Face Detection Approach

Human face detection is one of the classic applications of computer vision. Many algorithms based on computer vision have been proposed to detect human faces in still images and videos. Like human faces, animals of a particular class have faces with relatively fixed intra-class structure. Thus, a human face-detection algorithm can also be applied for the purpose of animal detection. Using this idea, a detection algorithm based on Haar-like features and AdaBoost, developed by Viola-Jones, is used to detect lion faces in videos, which achieved a detection rate of 93% [22]. A face-tracking model is implemented, using a Kanade-Lucas-Tomasi tracker to track the detected face. A specific interest model is then applied to the detected face and the two methods are combined in a specific tracking model. Thus, reliable detection and tracking of animal faces is achieved using this method [22].

1.3.3 Animal Detection Based on the Threshold Segmentation Method

Some methods have been developed to detect moving objects which are based on threshold segmentation. A moving object is detected based on background extraction which is divided into two steps, as moving object detection and background extraction. Object extraction from the background is performed by using a threshold segmentation method [23]. The threshold segmentation method based on pixel values evaluates the performance of object detection algorithms in video sequences. As the background image changes periodically, selection of an accurate threshold becomes difficult [24].

1.4 Objectives

The main objective of this thesis is to implement a system which can detect black-backed jackal faces (frontal) in images. To achieve this, the aim is to:

1. Survey available methods of animal detection.
2. Distil from the survey all that is necessary to implement an automatic system to detect black-backed jackal faces.
3. Implement Python code to develop a system which can differentiate between face and background using the Viola-Jones algorithm.
4. Construct a positive (black-backed jackal) and a negative background database.

1.5 Contributions

This thesis has the following outcomes:

1. An extensive literature study is presented on the different animal detection methods available. (Chapter 2)
2. A brief description of each key-feature of the Viola-Jones object detection algorithm is presented. (Chapter 3)
3. A comparative ROC curve is shown for Asymboost and Adaboost algorithm. (Section 4.5)
4. A system was implemented which in principle was shown to detect human frontal faces in still images. (Chapter 4)
5. Methods are presented for collection and construction of a large database of images. (Section 5.2, 5.3)
6. An additional Haar feature is used for BBJ and an improvement in accuracy is achieved over the basic set of Haar features. (Section 6.3.2)
7. A system to detect black-backed jackal faces in still images is implemented. (Chapter 6)

1.6 Overview

This study includes the detection of animals in still images. Animal detection is a well studied field, with different applications in various fields. Different methods have been developed

for the purpose of animal detection. Early research on animal detection comprises mostly of work towards finding how fast the human eye can detect the presence of animals in images. Said literature concludes that human eyes cannot function as efficient detectors as they easily become tired. This motivates the use of a computer vision algorithm for animal-detection tasks. Chapter 2 provides a brief literature study of available methods of animal detection.

Animal faces bear a close resemblance to human faces. Thus algorithms developed to detect human faces can also be applied for animal detection. The Viola-Jones algorithm is widely used for real-time object detection. The important key features of the Viola-Jones object detection algorithm are the fast feature evaluation of the Haar-features using integral images, feature selection using the Adaboost algorithm, and the cascade of strong classifiers. Chapter 3 provides a detailed discussion of each of the key features of the Viola-Jones framework.

The Viola-Jones object-detection algorithm was motivated in essence by the problem of human face detection. Thus a system was implemented to detect human faces in images. Various databases of human faces are available over the internet, of which the MIT-face database was used for training. Implementation details are discussed in Chapter 4 and a detector based on basic Haar features and an Asymboost algorithm is developed.

For the human faces, many databases are available over the internet. However, for the BBJ, it was necessary to create a database by collecting images of BBJ. Different methods for the collection of BBJ images and the preparation of a database of BBJ images are explained in detail in Chapter 5.

In this project, the basic aim was to develop a detector of black-backed jackal (BBJ) faces in images. A step-wise approach for the implementation of the jackal detection system is explained in Chapter 6. The chapter is concluded with a summary of the results and a discussion of the final detector. The final conclusion of the overall thesis and of future work is given in Chapter 7.

Chapter 2

Literature Review

2.1 Introduction

Animals play an important role in human life as a source of income or in maintaining an ecological balance. Animal-based research will also play an important role in the better management of animals along with human life. Animal detection, animal tracking, and the identification of animals are the three important applications in animal-based research [25]. Among these, animal detection is an important application due to its real-time applications in different fields. It includes mainly the use of technology that can detect larger-sized animals. Although much research on animal detection is mainly motivated by the problem of vehicle-animal collision, other important applications also include preventing the intrusion of wild or dangerous animals into residential or farming areas in order to save crops or livestock species.

First-branch animal detection is applied in various real-time applications. For example, a CVAAS (Camel Vehicle Accident Avoidance System) has been developed using GPS (global positioning system) technology for the automatic detection of the presence of a camel on or nearby the road in Saudi Arabia [26]. The research is mainly motivated to reduce the number of accidents caused by the animal-vehicle collisions (AVC), which result in the death of numerous people as well as loss of property. Airborne remote sensing images have been used to detect large mammals such as polar bears in snow [27]. Researchers [25, 28] developed a system using micro-Doppler signals to prevent the entry of dangerous animals into the residential areas to maintain public security.

Animal tracking includes the detection of animals and then the tracking of the detected

animal. The application of animal tracking includes the monitoring of locomotive behaviour and the study of the interaction of the targeted animal with the environment [25]. Researchers have developed a technology using sensors, RFID, and GPS to monitor an animal's behaviour. One important application of this technology includes the development of intelligent systems which can trace and identify an animal and provide real-time information about the current location, bodily temperature, and pictures of a targeted animal [29].

Animal identification includes the identification of detected animals. Animal identification helps humans to monitor and better manage their animals. It has been mainly used for animal care management in domestic herds. A mobile monitoring system (RFID-MMS) based on RFID has been developed to work over a wireless network to retrieve dynamic information and facilitate location tracking [30].

2.2 Animal Detection in Images and Videos

Animal detection usually includes the detection of larger mammals. Important applications of such systems include the prevention of animal-vehicle collisions (AVC) and the intrusion of wild animals into residential or farming areas, along with the study of the locomotive behaviour of targeted animals and the study of the role and habits of predators. Due to the vast number of applications in different fields, animal detection is an important field in image processing. Different methods have been developed to detect animals in images and videos. Some of these methods are summarised below.

2.2.1 Animal Detection by Human eyes

Most of the previous research was based on observing how fast and accurately the human eyes can perform animal detection in images. When the accuracy and speed is compared with computer vision algorithms, this method is considered the most reliable one [25]. Researchers use the go/no-go categorisation, in which an unseen image is flashed for 20 ms and the subject has to decide whether the image contains an animal or not [31]. They found that when natural unseen images are flashed for 20 ms, human eyes can detect the presence of an animal within 150 ms.

Although detection by the human eyes is accurate, this approach has major limitations.

The accuracy of human eyes depends mainly on distance and lighting factors. Human eyes can detect the presence of an animal if it is nearby and if there is sufficient lighting. Again, human eyes can easily tire, affecting accuracy and effectiveness. As they require rest, human eyes also cannot function 24 hours a day [25]. These limitations motivate the application of computer vision algorithms for animal detection tasks.

2.2.2 Animal Detection Based on Power Spectrum

The power spectrum of an image is constructed using Fourier transformation to transform an image from the spatial domain to the frequency domain. Object detection or animal detection in a scene or in an image can be predicted using the power-spectrum approach [25]. The idea behind this is to find whether the power spectrum of an image changes or not with the presence of an animal or an object in an image.

Researchers have worked on animal detection in natural images using simple image statistics. They used the statistical properties of natural images to categorise scenes and objects in images and achieved a rate of 80% for animal detection in natural scenes [32]. Findings also show that human observers use the approach of the power spectrum to locate animals in natural scenes and concluded that, for rapid animal detection, a human observer does use the power spectral difference between non-animal and animal images [33].

2.2.3 Animal Detection Based on Threshold Segmentation

Detection of moving objects in videos is one of the crucial tasks in image processing. Background subtraction is a common approach to enable detection of a moving object, which identifies an object that differs from the background [34]. This method faces challenges such as illumination changes, a moving background such as moving leaves and rain, and the shadow of an object.

Researchers used the threshold segmentation method based on pixel values [35]. Different object detection methods based on the segmentation algorithm, such as BBS (basic background subtraction), the W4 system, SGM (Single Gaussian Model) and LOTS (Lehigh Omnidirectional Tracking System) are evaluated to compare them and to evaluate the strengths and weaknesses of these methods [35].

2.2.4 Animal Detection Based on a Human Face Detection Approach

Animal faces bear a close resemblance to human faces. Human face-detection algorithms can thus be applied to the task of animal detection. Researchers applied a face detection algorithm based on Haar-features and Adaboost to detect animal faces in videos[36]. A detection algorithm based on Adaboost and Haar-features is introduced by Paul Viola and Michael Jones. The algorithm is applied to detect a lion face. Once a targeted animal is detected, a video recorder is turned on to track the detected animal. The applied tracking algorithm is based on the Kanade-Lucas-Tomasi method [36]. This method has the advantage that a person need not be present at the recording scene, thus dealing with the time and safety issue.

A neural network (NN) architecture has been developed using SVM as an inference engine to classify light detection and ranging (LIDAR) data to locate fish in deep waters [37]. A template-matching algorithm is used to detect animals (e.g. dogs) in videos [38]. The algorithm is divided into two steps: moving object detection and template matching. For moving object detection, a frame differencing method is used for background subtraction. Template matching is performed using the concept of normalized cross-correlation. ‘In signal processing, cross-correlation is a measure of similarity of two waveforms as a function of a time-lag applied to one of them. This is also known as a sliding dot product or sliding inner-product’. A low false positive and false negative rate was achieved by the proposed system.

The human face-detection algorithm shows good results [39, 40, 41] and should be investigated for animal detection tasks [36, 38]. This study emphasizes animal detection based on the detection algorithm for human faces. The human face-detection algorithms are described in more detail in the next section.

2.3 Human Face Detection

Image processing is used in various fields for tasks such as object detection, object recognition, visualization, and image sharpening and restoration. Object detection is one of the classic applications of computer vision in which the aim is to detect the object within a given image or video. Human face detection is a special case of object detection in which one aims to find

the location and size of faces within a given image [42]. Faces can be detected in images using facial features such as two eyes, one nose and one mouth. It is the very first and important step in many face-processing techniques such as face recognition, face location, face tracking, pose estimation, video surveillance and Human Computer Interaction, etc. The main purpose of each of various different face-processing techniques is given below.

1. Face detection: Aims to detect any face in a given image.
2. Face recognition: Aims to recognize/identify the face in a given image.
3. Facial expression recognition: Determines different emotional states of humans such as sadness, joy, excitement, etc. from the face in an image.
4. Face location: Returns the position of a face within an image.
5. Face tracking: Continuously provides the location of the face.

2.3.1 Challenges Faced by Human Face Detection

Human face detection is a challenging task, since faces in images are uncontrolled. Many algorithms have been proposed and improved, and yet face detection is beset with many problems due to uncontrolled factors present in images captured in uncontrolled conditions. The main challenges for face detection are described below [43].

1. Variant Pose: Subject's movement or a different camera angle leads to pose variation (Figure 2.1). The appearance of the subject can vary greatly with different poses, and the success of the face detection performance drops severely with large variations of pose.



Figure 2.1: Example of pose variation.

2. **Illumination Condition:** Different lighting conditions in images change the appearance of the face to a great extent. Figure 2.2 shows images with different illumination conditions. Different camera characteristics also affect the quality of images.



Figure 2.2: Example of different illumination condition.

3. **Facial Expression:** The face is a non-rigid object that changes with different expressions. Different expressions (Figure 2.3) on the face present different information to the machine.



Figure 2.3: Example of different facial expression.

4. **Uncontrolled Background:** Faces or object of interest in images are always located on a different background. Different backgrounds provide different information about objects in images. Thus, uniform information about the object of interest, or face, cannot be extracted from the images, which is one of the major issues in face detection.
5. **Face occlusion:** The presence of different objects such as sunglasses, hairstyle, hats etc. introduces great variability (Figure 2.4). Faces may also be occluded by other faces in an image containing more than one person.



Figure 2.4: Example of face occlusion.

2.3.2 Face Detection Algorithm

Face recognition is one of the more important research topics in image processing due to its wide variety of applications in law enforcement and numerous other purposes. For the recognition task, computation at pixel level is a time consuming and impractical approach.

Thus, to carry out face recognition and other techniques such as face tracking and face expression recognition, face detection is a primary need. More than 150 face- detection algorithms have been developed in recent years, which are well categorised by Yang [44] as Knowledge Based Methods, Feature Invariant Methods, Template Matching Methods and Appearance Based Methods. Early algorithms concentrated mostly on frontal face detection, whereas the new algorithms concentrate on multi-view face detection, appearance and lighting problems [42].

2.3.2.1 Knowledge Based Methods

Researchers have developed their algorithms based on the human knowledge of the fact that the human face contains two eyes, one nose and two ears. The distance between two features and their respective positions represent the relationship between features [44]. Even though this approach is simple, it is difficult to represent human knowledge in well-defined rules to carry out face detection. Strict rules may fail to detect faces, and more general rules increase the number of false positives. Again, this approach cannot be extended for multi-view face detection, since it is difficult to enumerate all possible cases [44]. Figure 2.5 shows a common face structure used to derive rules in knowledge-based methods. ‘Rules are coded based on human knowledge about the characteristics like intensity distribution and difference. The centre part of the face (dark shaded part in Fig. 2.5) has a uniform intensity while the upper round part of the face (light shaded part in Fig. 2.5) has uniform intensity. The difference between the average grey of the centre part and upper part is significant’ [44]. Yang and Huang developed an algorithm based on hierarchical knowledge of the human face that consists of three levels of rules to detect faces [1].

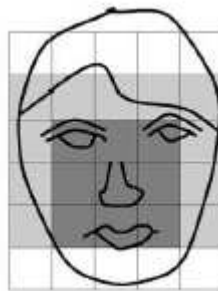


Figure 2.5: A common face template used in knowledge based methods [1].

2.3.2.2 Feature Invariant Algorithms

In this approach, researchers tried to find some facial features that are invariant in different poses and lighting conditions. Eyes, mouth, eyebrows, and hairline are some of the features of the face that remain unchanged with different lighting and pose conditions. These features are extracted using an edge detector, and their relationship is described using a statistical model. An algorithm is proposed by Sirohey [45] to identify faces in cluttered backgrounds using the localization method. A face model is proposed by Chetverikov and Lerch [46] using two dark blobs to represent two eyes and three light blobs that represent two cheekbones and a nose. Some of the algorithms use texture features such as human skin colour for face detection and hand tracking, etc. [47, 48, 49, 50]. A number of algorithms have been proposed for face detection and localization by combining several features such as skin colour, size and shape [51, 52]. One limitation of this method is that the facial features can become severely corrupted by the presence of noise and occlusion.

2.3.2.3 Template Matching Algorithms

In the template matching algorithm, a manually predefined standard face pattern is constructed. The correlation values for face contour, eyes, nose and mouth are computed independently with the standard face pattern. The existence of a face in any image is determined using these correlation values. Craw developed an algorithm based on the localisation method to design a shape template for the frontal face detection task [53]. Sinha used the idea of a set of spatial image variants to describe face patterns. The idea behind this is that the brightness of individual parts of the face, such as the eyes, cheeks and forehead, changes with different illumination [44]. A robust invariant is obtained by determining the pairwise ratios of brightness of such parts. Figure 2.6 shows the enhanced template with 23 defined relations. ‘These defined relations are further classified into 11 essential relations (solid arrows) and 12 confirming relations (dashed arrows). Each arrow in the figure indicates a relation, with the head of the arrow denoting the second region (i.e. the denominator of the ratio). A relation is satisfied for the face template if the ratio between two regions exceeds a threshold and a face is localized if the number of essential and confirming relations exceeds a threshold’ [44]. While this approach is simple, it cannot deal with different poses, scales and shapes of face. Further, deformable templates are

proposed for different poses and scales [54, 55, 56, 57, 58, 59].

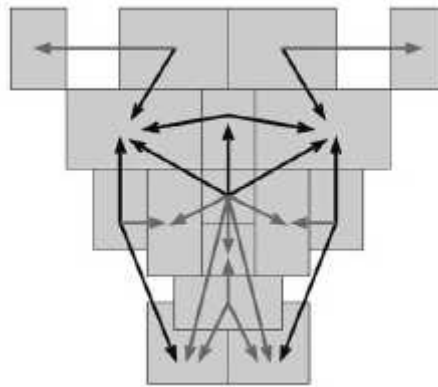


Figure 2.6: A 14*16 pixel ratio template for face localization based on the Sinha method. The template is composed of 16 regions (the grey boxes) and 23 relations (shown by arrows) [60].

2.3.2.4 Appearance Based Algorithms

Face detection algorithms that rely on statistical analysis and machine learning algorithms are categorised as appearance based methods. The models constructed use no prior knowledge about the face; instead the information about a face is extracted through the statistical analysis of a given training set, using machine learning algorithms. The trained model is then used to classify images as face or non-face. These algorithms usually contain two phases known as the training phase and the classification phase. The classification is done using classifiers such as a support vector machine (SVM), random forest (RF), principal component analysis (PCA) or eigenfaces, neural networks (NN), and boosting based classifiers such as Adaboost. Among the available methods of face detection, the appearance based methods show good results [44].

1. Support vector machine

‘The support vector machine is a supervised learning model with an associated learning algorithm which constructs a hyperplane or a set of hyperplanes in a high or infinite dimensional space which is used for classification purposes’ [61]. The SVM algorithm builds a model based on a set of training examples, each belonging to one of two classes. The model is then used to classify the input data and assign one of two classes, making it a non-probabilistic binary linear classifier. An SVM classifier is a linear classifier in which a hyperplane is chosen in such a way as to reduce the generalization error, since the larger the margin, the lower the generalization error [44]. Figure 2.7 shows a simple

classifier separating data into two classes, but the classifier line is passing too close to the data points; whereas Figure 2.8 shows an optimal hyperplane, generated by a SVM classifier that maximizes the margin of training data.

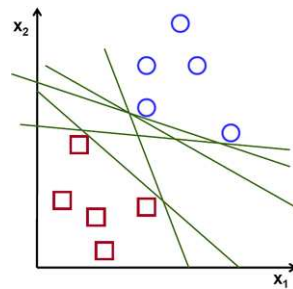


Figure 2.7: Simple straight line as simple classifier.

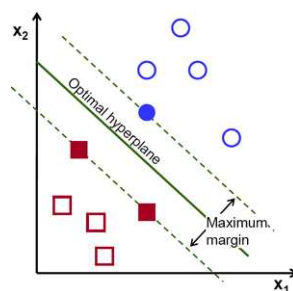


Figure 2.8: Optimal hyperplane as SVM classifier.

2. Principal component analysis

Principal component analysis (PCA) is a method used to identify the patterns in data of high dimensions. It was first introduced by Karl Pearson in 1901 [62]. PCA is mostly used for dimension reduction of data and for data analysis. In 1987 Sirovich and Kirby [63] used PCA on a dataset of face images to form a set of basis features. These basis features are known as eigenfaces and they are used for face recognition. Figure 2.9 shows examples of eigenfaces. PCA works by finding the representative vectors, called eigenvectors, corresponding to the largest eigenvalues of a covariance matrix in such a way that the projected samples retain most of the information about the original samples. A covariance matrix is constructed using eigenfaces, which is a basis set of all the images. In 1991 Turk and Pentland [64] furthered the idea of using eigenfaces for automatic face recognition. PCA is an example of an image based approach that solves face detection as a general pattern recognition problem, using statistical analysis and a machine learning algorithm. Instead of using features of the image, intensities of an example image are used to represent the characteristics which are then used for face detection [65].

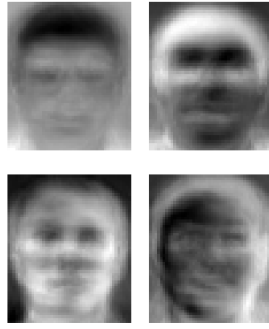


Figure 2.9: Eigenfaces

3. Random Forests

Random forest (RF) is a bagging based ensemble learning method used for classification and regression purposes. The algorithm for RF was first developed by Leo Breiman [4]. An idea of constructing a decision tree with controlled variation is the fusion of two ideas: bagging, proposed by Breiman, and random selection of features, introduced by Ho [66] and Amit and Geman [5]. RF constructs the forest of random trees (RT). The RT are identical to decision trees, but are trained with the different bootstrap samples of original data. For each sample, a classification tree is grown in which each node is split using the best split among all variables [2]. The classification is done by aggregating the predictions of the trees. The error rate is estimated using “out-of-bag” (OOB) estimates, i.e at each bootstrap iteration it is predicted whether the data is or is not in the bootstrap sample [2].

4. Boosting based method

In the boosting based method, the outputs of several weak classifiers are combined to form the final output. This allows for faster training in which each single weak classifier concentrates on a given portion of an image. For example, several neural networks (NN) are trained and the output of each neural network is combined with the others to form a final output. In 1989, Schapire proposed the first boosting algorithm. Figure 2.10 shows the concept of combining the weak classifiers.

Figure 2.10 shows that each weak classifier classifies an input x . The output of all the weak classifiers is combined to form the final classification decision. The final classification decision is given as:

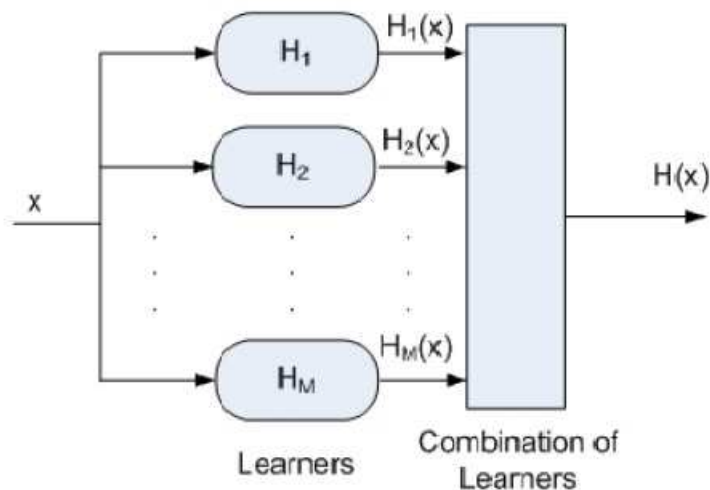


Figure 2.10: The concept of combining the outputs of the weak classifiers to produce the ensemble of classifiers [67].

$$H(x) = \text{sign} \left(\sum_{m=1}^M \alpha_m H_m(x) \right), \quad (2.1)$$

in which α_m is the weight of each weak classifier H_m . In 1996, Freund and Schapire [68] proposed the Adaboost (Adaptive Boosting) algorithm. The Adaboost algorithm [68, 69] is a well-known method of building ensembles of classifiers. Real Adaboost [70], Gentle Adaboost [49], Logitboost are some of the modified versions of Adaboost.

A boosting based algorithm has been proved to be a simple yet extremely effective algorithm for face detection purposes. Table 2.1 summarizes face detection methods that use boosting based algorithms [67]. A stub is defined as a decision tree with one decision node.

Table 2.1: Use of boosting algorithms for face detection.

Face Detector	AdaBoost Version	Weak classifier
Viola-Jones	Discrete AdaBoost	Stubs
Float Boost	Float Boost [71, 72]	1D Histograms
KLBoost	KLBoost [73]	1D Histograms
Schneiderman	Real AdaBoost [70]	One group of n-D Histograms

2.3.3 Applications of Face Detection

Face detection is used in various research fields such as public security, financial security and in human computer interaction for immigration management, automatic login system, virtual games etc.

Research areas that include applications of face detection:

1. Face recognition
2. Face image coding/compression
3. Face expression analysis
4. Emotion determination
5. Face attributes classification
6. Human computer interaction

2.4 Existing Face Detection Method

As we have seen so far, a number of promising face detection algorithms have been developed and proposed. Among these, three methods stand out since they are often referred to when performance figures, etc. are compared. In this section, the outline and main approach of each of these methods are described [74].

1. Robust real-time face detection, 2001

A face detection algorithm developed by Paul Viola and Michael Jones [41] in 2001 proved to be one of the most efficient face detection algorithms, with high accuracy and real-time application. Although it was primarily motivated by the problem of face detection, the algorithm can be used to detect a variety of objects. The main attributes are: use of an integral-image for Haar feature evaluation, use of the Adaboost algorithm for classification purposes, and the cascade of classifiers. The Viola-Jones face detection system classifies images based on Haar feature value, which is compared to the threshold set during learning. The Haar feature evaluation in real time is made possible by using an

integral-image. Haar features are combined to form a strong classifier using the Adaboost algorithm. The strong classifier generated by a learning process was still too inefficient to work in real-time; thus, strong classifiers are arranged in a cascade to form a complex structure, where each successive classifier is trained only on those samples which have passed through the preceding classifiers. This algorithm detects frontal upright faces, but in 2003 Viola-Jones presented a variant that detects profile and rotated views [75].

2. Neural network based face detection, 1998

This algorithm was developed by Rowley, Baluja and Kanade [39]. An image pyramid is constructed to achieve face detection at multiple scales. The basic principle of this algorithm is to run a sub-window of a fixed size through each layer in the image pyramid. The content of the sub-window is corrected in order to obtain histogram equalisation and non-uniform lighting. The processed data is then fed to several parallel neural networks (NN). The actual face detection is obtained by combining the outputs of each of the NNs using a logical AND. A previous version of this algorithm also detected only frontal upright faces.

3. A statistical method for 3D object detection

In 2000, Schneiderman and Kanade [40] developed an algorithm for the detection of 3D objects such as cars. This algorithm also follows the basic mechanics which form a pyramid of images. A fixed size sub-window is then scanned through each of the layers of the image pyramid. ‘The content of each sub-window is subjected to a wavelet analysis and histograms are made for different wavelet coefficients. These coefficients are fed to differently trained parallel detectors that are sensitive to various orientations of the object’ [74]. The orientation is determined by the detector that yields the highest output. This algorithm also detects profile views.

2.5 Summary

The basic problem of object detection is that the size and position of a given object within an image are unknown. Two of the above-mentioned algorithms [39, 40] overcome this problem by using a standard approach to calculate an image pyramid and scanning a fixed size detector

through each layer of the pyramid. However, this is time-consuming from an implementation point of view. Viola-Jones presents a novel approach to this problem by resizing the detector itself, rather than resizing the image, for multiple size face detection. A new representation of the image, called an Integral image, facilitates fast feature evaluation and a cascade of classifiers. These are the main components that enable this algorithm to run in real-time. This algorithm is simple to implement, offers a fast detection rate and high accuracy, and is one of the more widely used algorithms for object detection. The rest of this project will focus on the Viola-Jones face detection algorithm.

Chapter 3

The Viola-Jones Face Detector

3.1 Introduction

In 2001 Paul Viola and Michael Jones developed an algorithm which is one of the most widely used algorithms for real-time object detection [41]. Although Viola and Jones developed this algorithm for face detection, it can be used to detect a variety of objects such as cars, pedestrians, eyes, number plates, animals, etc. The system is trained from a large volume of positive and negative data. The positive data contains images of the objects of interest, such as faces, or cars. Negative data contains images of anything except the object of interest, i.e. background. The basic principle of most of the image processing techniques is to scan a fixed size detector through a pyramid of images, whereas in the Viola-Jones (V-J) technique a detector window capable of detecting faces is scanned through the image at different scales. To develop such a system, which can finely filter the false positives and detect the object of interest, the V-J technique uses three key features that make this possible in real-time. The first contribution is the new representation of an image, called an integral image. An integral image can be used to effectively evaluate Haar-features at any scale and location. Motivated by the work of Papageorgiou [76], the V-J system does not work directly with image intensities. Instead it uses Haar-features, which are similar to Haar basis functions. Another contribution of V-J system is the use of an Adaboost boosting algorithm for feature selection. AdaBoost is a simple and efficient classifier that selects small and important features (Haar features) from an extensive library. A single Haar feature is a set of type, size, location and parity, thus the number of Haar-features in any image sub-window is very large. In order to achieve fast

classification, a system should focus on small and critical features which can best distinguish between the object of interest and the background. Fast feature selection is achieved using an AdaBoost learning algorithm in which each stage of boosting selects a new feature. The third major contribution of this system is a cascaded structure of strong classifiers, which increases the speed of the detector by focusing attention on the part of an image which is likely to contain the object of interest. Each key feature is described in detail in the following sections.

3.2 Weak Classifier: Haar Features

Images are made up of pixels. A 24×24 image contains 576 pixels and working directly with the pixel intensity makes computation expensive. In addition, individual pixel data contains no information about the pixels around it. Thus, rather than working directly with the pixel intensity, Viola and Jones used Haar-like features which are similar to Haar wavelets. A Haar wavelet is a square shaped function with one high and one low interval [66]. Haar features are not true Haar wavelets; instead Viola and Jones use the combinations of rectangle better suited to the visual object detection task. That is why V-J called these features Haar features or Haar-like features. The idea of using Haar basis functions rather than working with pixel intensity was first proposed by Papageorgiou [76]. Haar features do not encode individual pixel information; instead they provide relative information on multiple pixels.

3.2.1 Upright Haar Feature

Viola and Jones used three types of upright Haar features, namely ones containing two rectangles, three rectangles and four rectangles, as shown in Figure 3.1. A feature value is the difference between the sum of the pixels within the rectangular region having the same size and shape, and which are vertically and horizontally adjacent. A large set of rectangle features are constructed from a particular type, with all possible size and location variations in a given sub-window of an image. At a base resolution of 24×24 , the exhaustive set of Haar features is quite large, namely 134736.

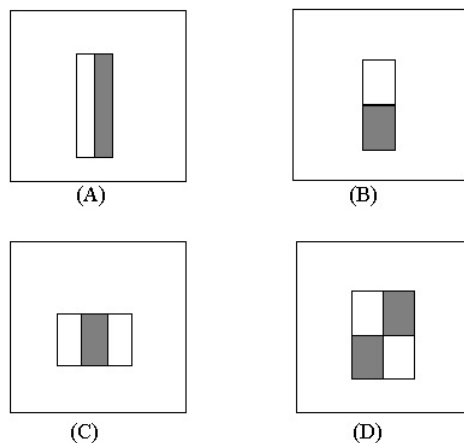


Figure 3.1: Four types of Haar feature, and two rectangle features are shown in (A) and (B). Figure (C) shows three rectangle features, and (D) shows four rectangle features.

3.2.1.1 Evaluation of Haar Features Using an Integral Image

The Viola-Jones face detection system classifies images based on the values of Haar features. The value of a Haar feature is the subtraction value of the average dark-region pixel value from the average bright-region pixel value. This difference is then compared with the threshold (set during learning); if the difference is above this threshold, that feature is said to be present. The feature value is calculated as,

FV = (sum of pixel intensities within dark region) - (sum of pixel intensities within bright region).

The rectangle feature is the difference of two box sums. A box sum is the sum of all pixel intensities within a given box. Let $I_n(x, y)$ represent the intensity of the pixel at (x, y) and $B(x, y, w, h)$ be the box sum of rectangle having width w , height h and (x, y) as the top-left coordinate. It is given as,

$$B(x, y, w, h) = \left(\sum_{x'=x}^{x+w-1} \sum_{y'=y}^{y+h-1} I_n(x', y') \right) \quad (3.1)$$

In order to compute the feature values rapidly, V-J introduce the concept of an integral image, which results in rapid computation of pixel intensities within a box and hence a rectangle feature. An integral image is the double integral of the image, (1) along rows, (2) along columns. The integral image at location (x, y) (bottom-right coordinate) is the sum of pixels to the left of (x, y) and above to (x, y) inclusive, as shown in Figure 3.3

Mathematically integral image is given as,

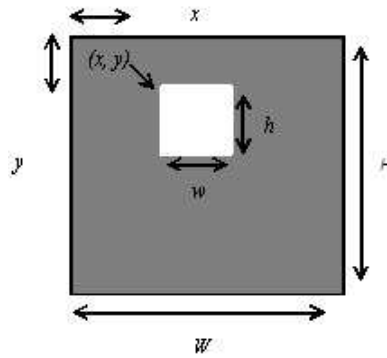


Figure 3.2: The features are constructed from the sum of pixel intensities within a rectangular region where (x, y) is the co-ordinate of its top left hand corner, w is the width and h is the height of rectangular region.

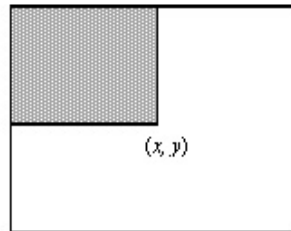


Figure 3.3: The value of the integral image at a point (x, y) is the sum of all the pixels above and to the left of that pixel.

$$II(x, y) = \left(\sum_{x' \leq x, y' \leq y} I(x', y') \right) \quad (3.2)$$

where $II(x, y)$ is the integral image and $I(x, y)$ is the original image. An integral image can be computed in one pass over the original image using equations 3.3 and 3.4 [62]:

$$S(x, y) = S(x, y - 1) + I(x, y) \quad (3.3)$$

$$II(x, y) = II(x - 1, y) + S(x, y), \quad (3.4)$$

where $S(x, y)$ is a cumulative sum over rows, given $S(x, -1) = 0$ and $II(-1, y) = 0$. The two rectangular features can be computed using six array references, eight in the case of a three-rectangle feature and nine in the case of a four-rectangle feature using an integral image. Figure 3.4 shows an original image and its integral image,

From Figure 3.3 it is clear that each pixel in the integral image is equal to the sum of all

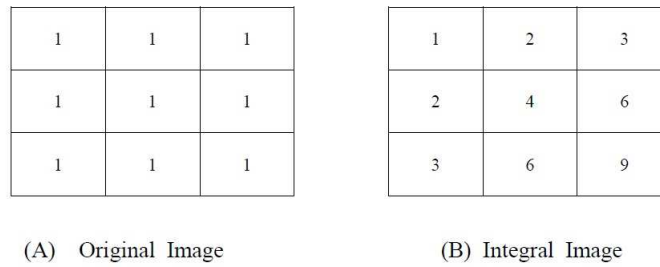


Figure 3.4: Original image and its integral image.

pixels that lie to the left and above that pixel. Thus using only four values, the sum of all pixels inside any given rectangle can be calculated. These values are the pixels in the integral image that coincide with the corners of the rectangle in the input image. This is illustrated in Figure 3.5,

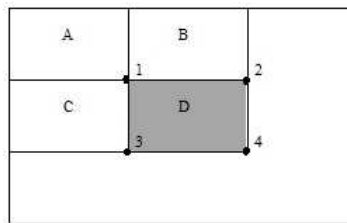


Figure 3.5: The sum within rectangle D can be computed as: $D = ii_4 + ii_1 - (ii_2 + ii_3)$.

where, $ii_1 =$ sum of pixel intensity in area ‘A’

$ii_2 =$ sum of pixel intensity in area ‘A’ + sum of pixel intensity in area ‘B’

$ii_3 =$ sum of pixel intensity in area ‘A’ + sum of pixel intensity in area ‘C’

$ii_4 =$ sum of pixel intensity in area ‘A’ + sum of pixel intensity in area ‘B’ + sum of pixel intensity in area ‘C’ + sum of pixel intensity in area ‘D’

In this way, an integral image is used to calculate the sum of pixels within any rectangles of arbitrary size. Before using Haar features to classify an object, the Haar feature values must be converted to true and false, and this is done using a threshold. Every Haar feature at a particular location and scale has its own threshold to classify objects.

3.2.2 Parity of Haar Feature

The direct Haar features has a specific polarity in terms of which sub parts of it will be weighed positively and which sub parts negatively. In general a detection system will find the use

of the inverse of such a feature useful, since this corresponds to recognising the reversed/inverted pattern of light and dark areas. This is achieved by extending the Haar feature with a parity of $+1$ or -1 . Multiplying the response of the image to a direct Haar feature with a parity of -1 , allows us to also simultaneously measure the response to the corresponding inverted Haar feature¹ (see Figure 3.6).



Figure 3.6: Direct Haar feature and its inverted counterpart.

3.2.3 Haar Feature Threshold

A Haar feature is the difference of the sum of pixel intensities between adjoining areas; it can thus be effectively used to detect edges in images. The zero value of a Haar feature indicates the lack of edge or the area having equal average intensity. To classify an object on the basis of Haar feature values, the value has to be compared with a threshold; if the value is greater than or equal to the threshold, the object of interest is said to be present.

$$h_j(x) = \begin{cases} 1 & \text{if } p_j f_j(x) \geq p_j \Theta_j \\ -1 & \text{otherwise} \end{cases} \quad (3.5)$$

where, $h_j(x) = j^{\text{th}}$ Weak classifier

$p_j =$ Parity of j^{th} Haar feature

$f_j(x) = j^{\text{th}}$ Haar feature value (with set size, shape and location)

$\Theta_j =$ Threshold of j^{th} Haar feature

Equation 3.5 illustrates that the output of the weak classifier is either 1 or -1 depending on the threshold [62]. The Haar feature threshold is evaluated in each round of boosting. For each feature, the feature value is evaluated relative to the same position in all the given images

¹In our implementation we added inverted Haar-features to the original set, thereby allowing us to ignore an explicit parity setting. Although potentially slightly less efficient, this is a numerically equivalent formulation.

which are cropped and resized. The example images are sorted based on feature values which act as a threshold. For each feature value, four sums are calculated as [41],

T_p = total sum of positive example weights,

T_n = total sum of negative example weights,

S_p = sum of positive weights below the threshold,

S_n = sum of negative weights below the threshold.

The error in respect of each feature value is evaluated as,

$$e = \min(S_p + (T_n - S_n), S_n + (T_p - S_p)) \quad (3.6)$$

The feature value with the minimum error is selected as the threshold for that feature. Each Haar feature has its own threshold and is updated in each round of boosting.

3.2.4 Discussion of the Haar Feature

Once the threshold of a Haar feature is determined, that Haar feature becomes a weak classifier and provides a Boolean result when placed over any portion of an image. A Haar feature is a set of type, size, location, parity and its own threshold. The pixels within the area of a Haar feature either sum to greater than the threshold or less than the threshold. Although a single Haar feature does a good job of detecting faces, it does produce numerous false positives. Thus a single Haar feature is termed a weak classifier. Therefore, in practice, a single Haar feature cannot be used for object detection.

3.3 Strong Classifier

As we have seen so far, a single weak classifier is not effective for the task of object detection. Thus, Viola and Jones developed the idea of forming a strong classifier in combination with a weak classifier, as described by Papageorgiou [76]. A strong classifier is the detector window loaded with weak classifiers (Figure 3.7), which is scanned across each image.

The standard approach of image processing is to scan a fixed size window and rescale the image. Contrary to this, V-J rescales the detector each time 1.25 times larger than the previous one [74]. A fixed scale window is then scanned across each of the images. V-J has empirically

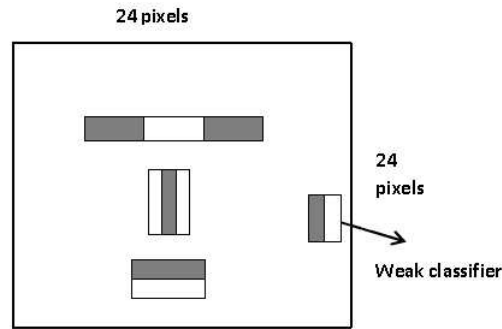


Figure 3.7: Strong classifier loaded consisting of several weak classifiers.

found that for human faces a detector window with the size of $24 * 24$ pixels gives a satisfactory result. A sub-window at its base resolution contains approximately 160000 Haar features. Computing a full set of Haar-features is quite time consuming and even impractical [41]. Hence to determine a set of the best weak classifier, Viola and Jones used a modified version of the boosting based algorithm known as AdaBoost, developed by Freund and Schapire in 1996 [68]. Further, Viola and Jones introduced the use of the AsymBoost algorithm, since AdaBoost minimises classification errors instead of minimising the false negatives.

3.3.1 AdaBoost

AdaBoost (Adaptive boosting) is a machine learning, boosting based algorithm capable of constructing a strong classifier from a weighted combination of weak classifiers. A weak classifier is expected to be correct in more than half of the cases, hence it is considered a potential weak classifier. Since only some of the features from the available features can be potential weak classifiers, 'the AdaBoost algorithm is modified to select only the best features' [74]. An algorithm not only selects the best feature, but is also used to train the cascade of classifiers. Figure 3.8 illustrates the AdaBoost algorithm.

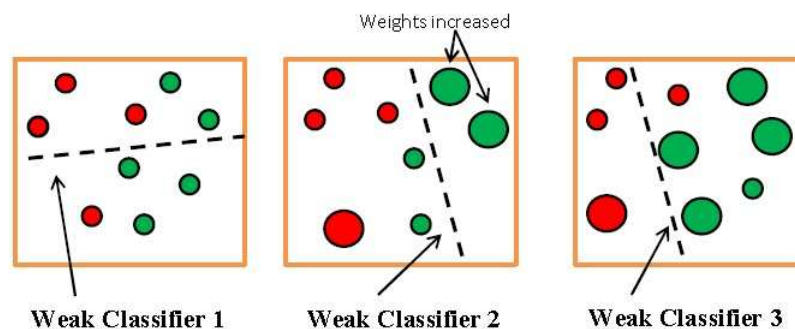


Figure 3.8: Classification using AdaBoost algorithm [2].

Input to AdaBoost is a set of Haar features and training examples. In each single round ($t = 1 \dots T$) of boosting, all features are evaluated on all training examples and the weighted error for each feature is determined. This weighted error is a function of the weights belonging to the training examples. The weak classifier with the lowest error is selected for a particular round. Once the weak classifier is added to the list, the weights of all the training examples are updated for the next round. The weight of misclassified examples is increased and the weight of correctly classified examples is decreased, as shown in Figure 3.8. The final strong classifier is a weighted sum of all the weak classifiers selected in that round. The AdaBoost algorithm for feature selection and learning is given in table 3.1.

Table 3.1: AdaBoost algorithm [4].

<ul style="list-style-type: none"> • Given example images $(x_1, y_1), \dots, (x_n, y_n)$ where $y_i = -1, 1$ for negative and positive examples respectively. • Initialize weights $w_{1,i} = \frac{1}{2m}, \frac{1}{2l}$ for $y_i = -1, 1$ respectively, where m and l are the number of negative and positive examples respectively. • For $t = 1, \dots, T$: <ol style="list-style-type: none"> 1. Normalize the weights, $w_{t,i} \leftarrow \frac{w_{t,i}}{\sum_{j=1}^n w_{t,j}} \quad (3.7)$ 2. Select classifier, h_t, with the minimum error ε_t . 3. Update the weights as, $W_{t+1}(i) = \frac{W_t(i) \exp \{-y_i h_t(x_i)\}}{Z_t} \quad (3.8)$ <p>where,</p> $Z_t = \sum_i W_t(i) \exp \{-y_i h_t(x_i)\} \quad (3.9)$ • The final strong classifier is: $h(x) = \begin{cases} 1 & \sum_{t=1}^T \alpha_t h_t(x) \geq \frac{1}{2} \sum_{t=1}^T \alpha_t \\ -1 & \text{otherwise} \end{cases}$ <p>where $\alpha_t = \log \frac{1}{\beta_t}$ and $\beta_t = \frac{\varepsilon_t}{1-\varepsilon_t}$</p>

3.3.2 AsymBoost

In AdaBoost, each round of boosting selects a feature with a minimum error. The selection of features and the training proceeds in such way as to minimise classification errors. This limits the usefulness of AdaBoost, as it does not reduce the number of false negatives. To overcome this problem, Viola-Jones introduce AsymBoost (asymmetric AdaBoost) [63] in which more weight is given to the positive examples compared to negative examples. Following the notation of Shapire and Singer [63], the examples are labelled as $\{-1, 1\}$ with -1 for the negative and 1 for the positive images. The important difference between AdaBoost and AsymBoost lies in weight updating. In AdaBoost, the weights of examples are updated using [4],

$$W_{t+1}(i) = \frac{W_t(i) \exp \{-y_i h_t(x_i)\}}{Z_t} \quad (3.10)$$

while in AsymBoost weights are updated as,

$$W_{t+1}(i) = \frac{W_t(i) \exp \{-y_i h_t(x_i)\} \exp \{y_i \log \sqrt{k}\}}{Z_t} \quad (3.11)$$

where k is the level of asymmetry and Z_t is used to normalise the weights of examples. While weight updating in AsymBoost, the term $\exp \{y_i \log \sqrt{k}\}$ adds more weight to positive examples in each round of boosting, since false positives cost k times more than false negatives. In this project k is set to 1.5. Thus AsymBoost solves the problem of asymmetry which is an intrinsic part of the learning procedure in AdaBoost.

3.4 Cascading Strong Classifiers

To increase the computational efficiency of their method, Viola and Jones use a cascade of classifiers. Cascaded classifiers are a cascade of strong classifiers. The idea behind using the cascaded classifier is that it is faster to reject non-faces than it is to find faces. For this purpose a detector with only one strong classifier seems inefficient, hence the need arose for a cascade of classifiers. Viola and Jones also refer to the cascade classifier as an ‘attentional cascade’, since more attention is given to the region of the image suspected of containing faces.

The strong classifier in a cascade is trained using an AdaBoost/AsymBoost algorithm. In this project we use the AsymBoost algorithm to train strong classifiers in a cascade in order

to reduce false negatives. The first stage of the cascade contains the smallest number of weak classifiers, requiring the smallest number of computations. ‘The job of each stage is to determine whether a given sub-window is definitely not a face or may be a face’ [74]. If a given stage identifies a sub-window as a face, that sub-window is passed to the next stage, otherwise it is discarded as a non-face. ‘It follows that the more stages a given sub-window passes, the higher the chance that the sub-window actually contains the face’ [74]. There are many false positives in the initial stages of a cascade, but these eventually decrease as the number of strong classifiers in the cascade goes on increasing. Figure 3.9 illustrates the cascade of classifiers.

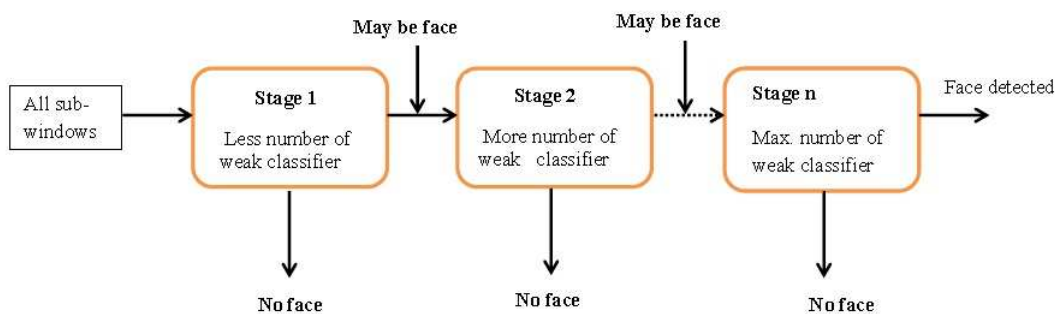


Figure 3.9: The Cascade of classifiers.

As shown in Figure 3.9, stage 1 contains a smaller number of weak classifiers than does stage 2, and the number of weak classifiers increases up to stage n . Input to stage 1 is all the possible sub-windows of an image; if stage 1 identifies the given sub-window as a face, the sub-window is transferred to the next stage, otherwise it is discarded as non-face. As each sub-window passes more stages, there is a greater possibility that a given sub-window contains a face. Thus, using a cascaded classifier, more false positives can be reduced with an increase in the number of face detections. The cascade learning algorithm is shown in Table 3.2.

The initial threshold $\left(\frac{1}{2} \sum_{t=1}^T \alpha_t\right)$ described in [41] is designed to achieve a low error rate on a training set. The threshold is adjusted in such a way as to achieve a higher detection rate with a minimum of false positives. The false positive rate of a trained cascade classifier is given as [41]:

$$F_p = \left(\prod_{i=1}^M F_i \right), \quad (3.12)$$

where F_p is the false positive rate of the cascaded classifier, M is the number of classifiers, and F_i is the false positive rate of the i th classifier. The detection rate is given as:

Table 3.2: Cascade learning algorithm [5].

- Given P a set of positive examples, N an initial set of negative examples, D a database of negative examples. Given false positive rate goal F_p for the cascade. Output is cascade classifier $H = (H_1, H_2, \dots, H_r)$.
- $i = 0$;
- While current cascade's false positive rate is bigger than F_p .
 1. $i = i + 1$.
 2. Use AsymBoost to train a classifier H_i with P and N . Add H_i to cascade H ;
 3. $N' = \phi$. Run H on D , add false positives (negative examples classified as positive by H) to N' , until $|N'| = |N|$.
 4. $N \leftarrow N'$.

$$D_t = \left(\prod_{i=1}^M D_i \right), \quad (3.13)$$

where D_t is detection rate of the cascaded classifier, M is the number of classifiers, and D_i is the detection rate of the i th classifier.

3.5 Variance Normalisation

Differing illumination in images severely reduces the detection rate of a system. Thus, to compensate for different lighting conditions, Viola and Jones normalised all training images. Variance normalisation of each training sample is obtained by determining the variance of the pixel values in the detector window as $\sigma^2 = \left(\frac{1}{n_p} \sum x_p^2 - \mu^2 \right)$, where σ^2 is the variance, n_p is the total number of pixels in the detector window, x_p is a pixel value and μ is the mean of the pixel values in the detector window. Then each pixel is normalised as,

$$x'_p = \frac{x_p - \mu}{c\sigma} \quad (3.14)$$

where σ is the standard deviation. In this project c was set to 2. The standard deviation can

be calculated by taking the square root of variance (σ^2). Recalculating every pixel in every sub-window cannot be done in real-time. However, the mean μ of a sub-window can easily be found using the integral image. The total pixel sum of a sub-window can be obtained directly from the integral image, divided by the number of pixels we get from μ . In order to obtain the sum of squared pixels, Viola and Jones introduced a squared integral image. The squared integral image is an integral image of the squared image. While the system was trained using normalised samples, during detection the samples would also have to be normalised and this is achieved by ‘post-multiplying the feature values, rather than pre-multiplying the pixels’ [64].

3.6 Scanning of a Detector Window

As discussed before, a detector is a sub-window of a specific size loaded with weak classifiers. During detection, a detector window is scanned across the image at multiple scales and locations. The input image is first converted to an integral image and a detector window is run across the image. The image is scanned n times; each time the detector window is larger by a factor of $s = 1.25$. Rather than scaling the image, Viola and Jones scaled the detector itself, since the features can be evaluated at any scale using an integral image.

The detector is also scanned across different locations with a number of pixels (Δ) . For the current scale s , the window should be shifted by $[s\Delta]$, where $[]$ is the rounding operation. The time and accuracy depend on the shifting of the window [64].

3.7 Integration of Multiple Detections

The final detector is often insensitive to small variations in translation and scale. This results in multiple detections around a face, which can be quite confusing to work with. Thus, the multiple detections are merged to return one final detection [41]. This is achieved by following two algorithms. In the first algorithm, the detections are merged if their centres coincide. In the second algorithm, the overlaps are merged if they overlap more than 25%. Merging overlaps also decreases the number of false positives and provides one final detection per face.

3.8 Summary

The important key features of the Viola-Jones object detection algorithm are the fast feature evaluation of the Haar features using integral images, feature selection using the Adaboost algorithm, and the cascade of strong classifiers. The Viola-Jones algorithm used the integral image for the evaluation of Haar features at any scale and location. The images to be used for the training are first converted into greyscale, and variance is then normalised. The Adaboost algorithm selects the Haar features with the minimum number of errors in each round of boosting. To minimise the false positive rate in real-time, the strong classifiers are arranged in a cascade using the cascade learning algorithm. The testing phase of the Viola-Jones system includes the scanning of a detector window and the resizing of the detector window, each time by a particular factor. The testing images are also variance-normalised to ensure a high detection rate. Once the object of interest is detected in an image, the multiple detections that occur around a face are finalised to a single detection.

Chapter 4

Human Face Detection

4.1 Introduction

This chapter presents the implementation of the Viola-Jones object detector to detect human faces. The algorithm is expected to find the faces in an image if there are any, and Figure 4.1 shows an example image. The aim was to develop a system that uses the Viola-Jones object detection algorithm to locate faces in images. The primary requirement of the algorithm is the human face database, since the system is trained on the basis of the features that are present in the training images. Training images comprise both the positive, i.e the human face, and the negative, i.e background images. Prior to input of the images into the algorithm, the images need to be pre-processed to ensure a high detection rate. The implementation and preparation of positive and negative data/samples is explained in the following section.



Figure 4.1: Example image for human face detection.

4.2 Preparation of Database

The preparation of a database of objects of interest can be done by downloading images from the web, extracting them from videos and taking photographs of objects of interest. The images collected from various sources cannot be used as they are, since the images contain a great deal of background and noise. Therefore the images are pre-processed in order to create a useful database. The first step of pre-processing includes the conversion of colour images into greyscale. Prior to using the images for training, the Viola-Jones technique requires that the images be converted to greyscale in order to reduce the amount of information they contain. Grayscale conversion of images is explained in detail in section 5.4.1. Once the images have been converted to greyscale, an object of interest is cropped from the image and resized to a particular scale. Finally all the cropped and resized images are variance-normalised to compensate for the different lighting conditions. Viola and Jones suggest this pre-processing to ensure a high detection rate. A detailed explanation of image-processing is given in the next chapter.

4.2.1 Available Human Face Databases

In the past decade, face detection has been one of the most studied topics in image processing. Consequently, various human face databases can be found on the internet. Some of these are discussed below.

1. FERET: This database contains 14051 images of 1000 people comprising both male and female faces. The database was created by the FERET program from 1993 to 1997.

See:<http://www.itl.nist.gov/iad/humanid/feret/>



Figure 4.2: Example images from FERET database.

2. BioID: This database contains 1521 images of 23 people. The images are greyscale and show a frontal view of each face. The faces in the images are of different sizes.

See:<http://www.bioid.com>



Figure 4.3: Example images from BioID database.

3. CMU face database: This database is composed of four sets of testing images. Each set contains face and non-face images of different sizes. It has 130 images with 511 frontal views of faces.

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



Figure 4.4: Example images from CMU database.

4. MIT Face Database: The MIT face database contains two sets: one for training and one for testing. The training set contains 2429 face images and 4548 non-face images, whereas the testing set contains 472 faces and 23573 non-faces. Each image is in greyscale and has been rescaled to $19 * 19$.

See: <http://cbcl.mit.edu/software-datasets/FaceData2.html>

In this project we used the MIT face database, since each image in this database is cropped and resized to $19 * 19$, which thus reduces the effort of image processing. Again, the database contains a sufficient number of images for training purposes. Figure 4.5 shows the face and non-face example images from the MIT face database.

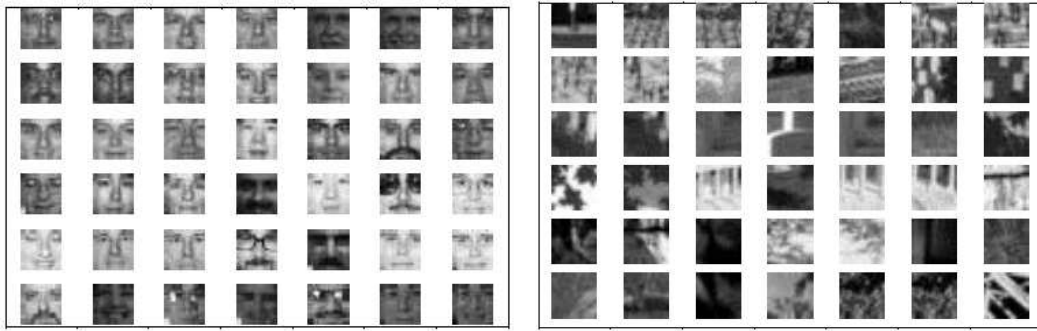


Figure 4.5: Face and Nonface example images from the MIT-database.

4.2.2 Negative Database

For training of the different stages of the cascade, negative or background samples are required. Basically, negative samples are the images not containing a face. In order to train different stages, different negative images are required. Also, the negative examples should represent all sorts of non-facial textures. Thus, to create a large database of negative samples, patches from the images are cropped that are guaranteed not to contain a face. The negative database for human face training includes a wide range of example images, and Figure 4.6 shows some of the images used to generate negative samples for a human face detection training system.

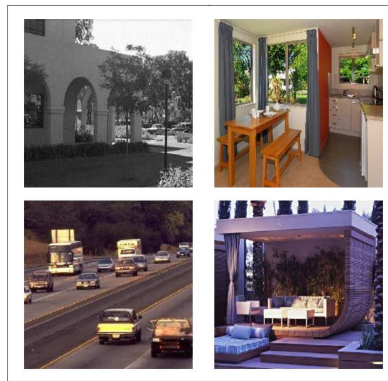


Figure 4.6: Background images for human face detection system.

4.3 Implementation of the Human Face Detection System

The human face detection system was implemented using the Python programming language. The process of developing a detection system starts with introducing a set of Haar features and training images to a boosting algorithm. The training images are first pre-processed. The

training process is speeded-up by using multiprocessing to select the best Haar feature of each type. The implementation of the detection system includes the steps outlined below.

4.3.1 Pre-Processing of the Image

The database of positive and negative images contains the cropped images. However, these images need some pre-processing before being input into the algorithm. The code for the pre-processing of the images includes the following steps:

1. Open an image and resize it.
2. Convert the image to greyscale.
3. Perform variance normalisation of the image.
4. Save the variance normalised file.

4.3.2 Implementation

To implement the detection system, we need a Haar feature file and positive and negative samples. As we have already created both positive and negative samples, the aim is to create a Haar feature file. A single Haar feature is a set of (x, y, w, h, f, p) , where x, y is the location, w, h is the size, f is the feature type and p is the parity of the Haar feature. A basic set of Haar features, as shown in Figure 3.1, is used to train the system, since the system is implemented to detect faces displayed from the front and the right way up. The Haar feature file contains all Haar features of each type. The Haar feature file is created by the following process:

1. Open an image.
2. Get Haar feature of type A at all possible locations with all possible sizes.
3. Get Haar feature of type B at all possible locations with all possible sizes.
4. Get Haar feature of type C at all possible locations with all possible sizes.
5. Get Haar feature of type D at all possible locations with all possible sizes.
6. Save all features of types A, B, C and D in a file named Haar features.

Table 4.1: Relationship of number of Haar features to the particular size of a window.

Size of Window	Number of Haar features
19 * 19	53130
24 * 24	134736
30 * 30	327300
36 * 36	676404

Table 4.1 shows the total number of Haar features in a particular size of window. As the size of the training images used is 19 * 19, the total number of Haar features is 53130.

A single strong classifier is trained using a boosting algorithm [41, 63]. The positive images, negative images and a Haar feature file are input to the algorithm. To accelerate the training process, multiprocessing is implemented for feature selection. In each round of boosting, a single weak classifier is selected and added to the list. The output obtained is a strong classifier. A strong classifier is a file containing weak classifiers, the weight of each weak classifier, and a threshold for weak classifiers. Once the strong classifier is obtained, it is evaluated against validation data to check the false positive rate of the system. Each strong classifier in a cascade is trained using positives and a separate set of negative images that pass through the previous stage. The cascade of classifiers is trained using an algorithm described in Table 3.2. The strong classifier is added to the cascade until the desired false positive rate is achieved.

4.4 Tools Used in Project

The human face detection system was developed using the following tools:

4.4.1 Hardware

1. Processor: Intel Core i5-2450M CPU @ 2.50GHz 4
2. Memory: 7.7 GiB

4.4.2 Software

1. Linux OS

2. Python 2.7
3. OpenCV

4.5 Experiments

4.5.1 Experiment 1:

In a paper [63], Viola and Jones proposed an Asymboost algorithm to decrease the number of false negatives. According to Viola and Jones, the feature selection process using an Adaboost algorithm proceeds in such a way as to reduce the classification error instead of rejecting negative samples. Thus, Viola and Jones used an algorithm that modifies the initial distribution over the training examples, in which false negatives cost k times more than false positives.

1. Purpose: Here the experiment was performed to determine whether the Asymboost algorithm performs more accurately than the Adaboost algorithm when the two systems are developed with the same complexity.
2. Execution: We trained two systems using the Adaboost and Asymboost algorithms. The two algorithms were written using Python code. The training set consisted of 2429 face images and 4548 non-face images. The systems were developed using a basic set of Haar features. Two systems were trained in such a way as to form a cascade of four strong classifiers. Each strong classifier contained four weak classifiers. Each strong classifier in a cascade was trained using positives and false positives that had passed through the previous stage.
3. Results: The two systems were evaluated on the same testing dataset consisting of 472 face images and 5000 negative images of size $19 * 19$. As the two systems are not fully trained systems, the images in the testing set are $19 * 19$ in size. Figure 4.7 shows the ROC curve of the two systems.
4. Interpretation of results: After the evaluation of two systems on the testing dataset, it was found that, for the false positive rate of 0.25, the system trained using an Asymboost algorithm achieved a detection rate of 0.89, whereas the system trained using an Adaboost

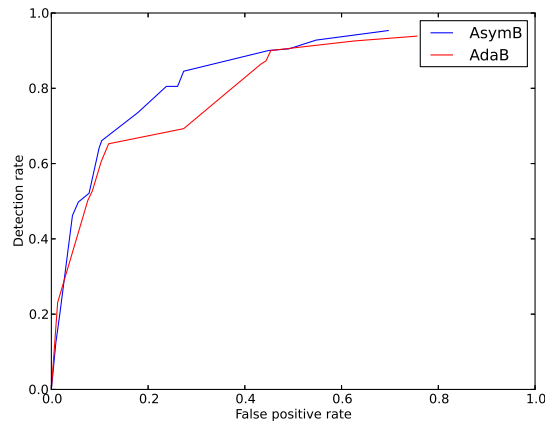


Figure 4.7: ROC curve showing comparison between Adaboost and Asymboost systems. The system was evaluated on 472 face images and 5000 negative images of size $19 * 19$.

algorithm achieved a detection rate of 0.69. This clearly implies that the Asymboost algorithm provides a higher detection rate than the Adaboost algorithm. On the other hand, it can be said that an Asymboost algorithm works better to lower the false negative rate compared to Adaboost when the two systems are of the same complexity.

4.5.2 Experiment 2:

1. Purpose: The results of experiment 1 show that the Asymboost algorithm performs more accurately than the Adaboost algorithm when both systems are of the same complexity. The purpose of this experiment is to find whether the Asymboost system also shows better accuracy over the Adaboost system when the two systems are fully trained. In addition, the experiment was designed to check the reliability of Python code by comparing an Adaboost system with a system trained using OpenCV-Haartraining.
2. Execution: We trained two systems based on Python code using the Adaboost and Asymboost algorithms. The training set comprised 2429 images of faces, 4548 non-face images, and approximately 1 million negative images for training different stages in a cascade. Both systems were evaluated on the same validation dataset. A strong classifier was evaluated on a validation dataset comprising 472 images of faces and 5000 non-face images to check the false positive rate of the systems. The classifiers were added to the cascade unless a false positive rate of 0.08 was obtained on the validation dataset. The desired false positive rate was achieved by completing the 14 stages of Adaboost system,

whereas for Asymboost system it was achieved with 16 stages.

In addition a 14-stage system was trained using OpenCV-Haartraining with a hit rate of 0.99, a false positive rate of 0.5, and using the same training dataset used to train an Asymboost and Adaboost system. An OpenCV-Haartraining is explained in more detail in Appendix A.

3. Results: The three systems were evaluated on the same test data set, containing 162 front-view face images from the BioID face dataset. Figure 4.8 shows the ROC curve of the three systems.

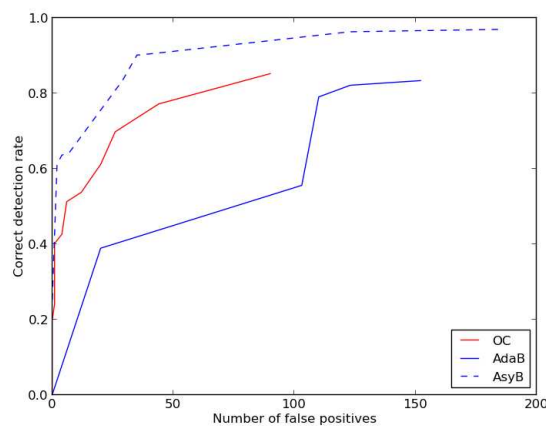


Figure 4.8: ROC curve for Adaboost, Asymboost and OpenCV systems. All three systems were evaluated on 162 images from the BioID test set. The figure shows that the Asymboost system achieves a higher detection than the Adaboost system.

4. Interpretation of results: The ROC curve in Figure 4.8 shows that the Asymboost system achieves greater accuracy than the Adaboost system when the two systems are trained at their best attainable performance. The Asymboost system reduces false negatives more severely than the Adaboost system. It also achieves a higher detection rate with a lower number of false positives. It can also be concluded from the Figure 4.8 that the system trained using OpenCV-Haartraining shows much better results than the Adaboost system trained using Python code. The possible reason for this is that the OpenCV is a highly refined code optimised by many people over several years. In all likelihood the system developed using Python code is simply not as polished.

The primary aim of this experiment was to check the accuracy of the Asymboost algorithm over the Adaboost algorithm when the two systems are performing at their best attainable

levels.

4.6 Final Face Detector

In this project the final face detector is a detector trained using an Asymboost algorithm with the basic set of Haar features. Each classifier in a cascade was trained on a separate set of negatives. The final face detector consists of 16 stages. The final detector was evaluated on a full set of BioID face database, which contained 1521 images. All the images were of the same size and contained front-view faces of different sizes. In addition, the images in the database had a variety of illumination levels and backgrounds. The BioID database can therefore be used effectively to evaluate the face detector. The threshold of the final stage of a detector is adjusted for the maximum detection and minimum false positives, so that the detector should perform at its best. The starting scale of a detector was $48 * 48$ with an increment of 1.25, and a step size of 2 pixels. According to Viola and Jones, the starting scale should be the size of the images used for training. Since the BioID face database does not contain any faces of small size, the starting scale was taken as $48 * 48$. Furthermore, using a starting scale with prior knowledge of the face size in the input image also reduces the number of false positives. Once the detector had marked the detection over an image, the post-processing was done to merge overlaps as discussed in section 3.7.

4.7 Results and Discussion

Figure 4.9 shows the faces marked by the detector. Among 1521 faces, the detector detected 1340 faces. The total number of windows scanned by the detector was 132177942, among which there were 1349 false positives. Thus, the detector achieves a detection rate of 88% ($1340/1521$) with a false positive rate 0.001% ($1349/132177942$). The detection rate of the system can be increased by decreasing the threshold of the final stage, but this will also increase the number of false positives.

We also evaluated the detector on the CMU-test set. The images in this database are of different sizes and levels of illumination. The faces in the images are also of different sizes. The detector was evaluated on three sets. The fourth one, a rotated test set which contained

rotated faces, was excluded as the detector was trained on front-view faces only. Figures 4.10 and 4.11 show the example images from the CMU-test set. Figure 4.10 shows that the detector works well when there are individual faces in an image or when there is sufficient difference between two faces. However, the algorithm is less successful when there are many faces in an image. The reason for this could be that the faces are not well separated.

Finally, it can be concluded that the false positive rate of the final detector is still rather high for the system and that the system needs to incorporate more stages to decrease the false positive rate. Again, it can be concluded that the detector works for images of a certain resolution, and the detection rate of the system gradually decreases for lower resolution images.



Figure 4.9: An example of images from the BioID test-set.

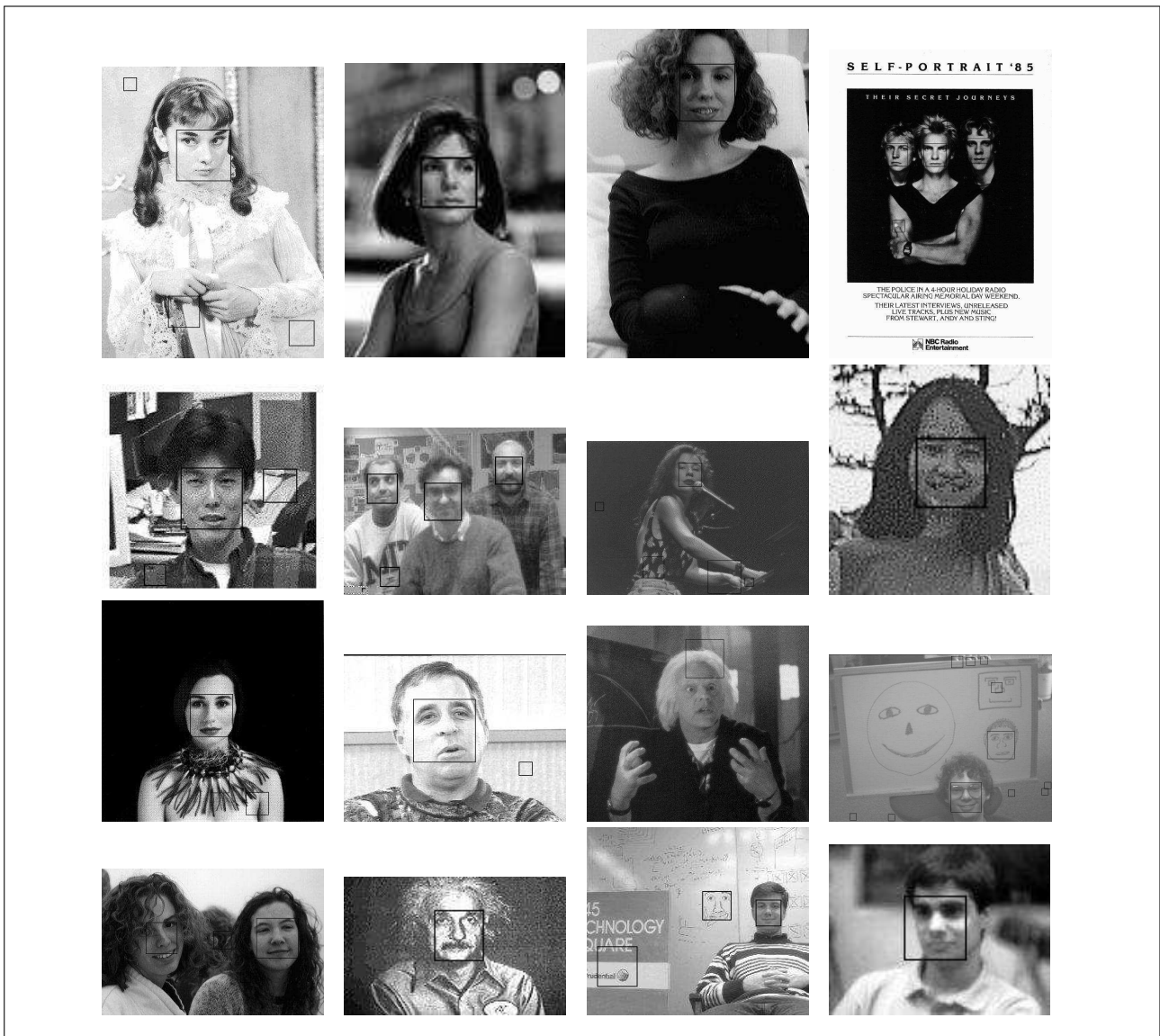


Figure 4.10: A - An example of images from the CMU test-set.



Figure 4.11: B - An example of images from the CMU test-set.

4.8 Summary

This chapter has presented the implementation of a human face detection system. As human face detection is one of the most widely studied aspects in the field, various databases of human faces are available over the internet. A short summary of available databases is given in section 4.2.1. A human face detection system was implemented using the MIT face database, since the images are already cropped and resized. Different negatives required for training different stages of a cascade are obtained by cropping patches from the images that are guaranteed not to contain any faces. The pre-processing needed for the images before introducing an image into an algorithm is explained in section 4.3.1. The implementation of the human face detection system is done using Python. In addition to this, a system is also trained using an OpenCV-Haartraining.

An experiment (section 4.5.1) comparing the Adaboost and Asymboost algorithms concluded that an Asymboost system reduced the number of false negatives more efficiently compared to an Adaboost system when trained with same complexity. The ROC curve in an experiment (section 4.5.2) shows the comparative result of three systems, an Asymboost and an Adaboost system trained using Python, and a system trained using an OpenCV-Haartraining. Finally, the chapter ends with discussion and results produced by the Asymboost system.

Chapter 5

Jackal Database Preparation

5.1 Introduction

In this project, the basic aim was to implement a system based on the Viola-Jones object detection algorithm to detect black-backed jackal (BBJ) faces in images. In the previous chapter, an algorithm to detect human faces was implemented. For the implementation of the algorithm, the basic need is for the training of images showing the object of interest. For the human face, many databases are available over the internet. On the other hand, for the BBJ, it was necessary to create a database by collecting images of BBJ. In this project, the aim was to detect the frontal faces of BBJ, thus the need to collect the frontal face images of BBJ. The following section explains the preparation of the BBJ database.

5.2 Collection of BBJ Images

The images of BBJ were collected by using the methods below.

5.2.1 Downloading from the Internet

The internet is the main resource for images of any type. Thus our collection of BBJ images included the internet as the main source. A number of images of BBJ are available over the web. The main criteria for the selection of images were that an image should be of the frontal face, and of high enough resolution. Using these criteria, a number of images was downloaded from the internet. The advantage of this technique was that it was an easy, free and quick way

of collecting images. Also, a variety of images of an object can be collected from one resource. Figure 5.1 shows examples of images collected from the internet.



Figure 5.1: Images collected from the internet.

5.2.2 Taking Photographs with a Camera

Taking photographs is one of the methods used to collect images of the object of interest. The images of BBJ were collected by taking photographs of BBJ in the field. A number of photographs were taken using a camera, but as the research required the frontal face of BBJ, we were able to add few images by way of this technique. From the collection of photos, images with the frontal face view were selected. The advantage of this technique was that it provided high-resolution pictures by using a good camera. The disadvantage of this method was the need to spend a great deal of time in taking photographs. Figure 5.2 shows examples of images collected using a camera.



Figure 5.2: Images taken with a camera.

5.2.3 Extracting from Videos

Extracting images from videos is a good method for the collection of large numbers of images of the object of interest. In order to extract the images of BBJ, we downloaded videos of BBJ available over the internet. From these videos, frames were extracted which provided a number of images of BBJ. The advantage of this method was that a number of images could be obtained at a time. This method also had the disadvantage that the resolution of images depended on the quality of the video. Also, many such images are very similar, which could cause the algorithm to be focused on object identification rather than object detection. This method also had the limitation that very few videos of BBJ were available and the resolution of the videos was also poor. For this reason, it was possible to add only a few images to the collection using this method. Figure 5.3 shows examples of images extracted from videos.

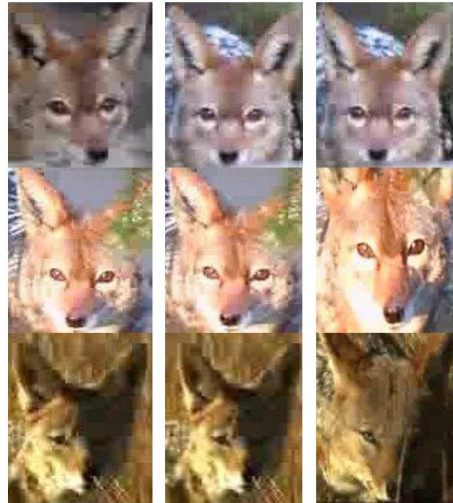


Figure 5.3: Examples of images extracted from videos.

5.3 Preparation of Database

The previous section explains how the images of BBJ were collected from different sources. Using these methods, 747 images of BBJ were collected. This database of BBJ was divided into three sets, namely a training set, a validation set, and a testing set. We used 510 images for training purposes, whereas 100 images were used for validation and 137 images were used for testing purposes. According to Viola and Jones, a greater number of training images increases the accuracy of the system. Thus the researcher strove to create a greater number of images from those available. The following section explains the method used to create more images

from the available ones.

5.3.1 Generating Positive Samples

The number of positive training samples was increased by using two methods: image rotation and image flipping. For this purpose, the basic set of training images, i.e 510 original images, was used. The methods are described below:

1. Image Rotation

The positive database of BBJ was increased by slightly rotating the images. This technique creates variation in images and allows one to create a greater number of images from those available. In order to achieve the required rotation, images are rotated by 3° , 6° , -3° and -6° . Figure 5.4 shows examples of the images created by rotation.



Figure 5.4: Original image and image rotated by 6° .

2. Image Flipping

The number of images in the positive database can also be increased by way of image flipping. The horizontal flipping of images provides images with some variation, but this technique is used for some images, as frontal face images were needed for the BBJ database. Figure 5.5 shows examples of images created by way of horizontal flipping.

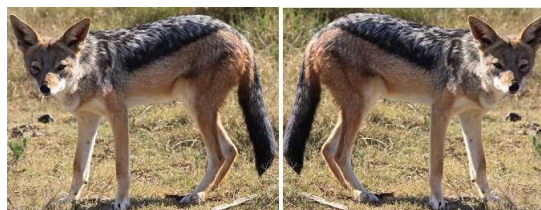


Figure 5.5: Original image and flipped image.

Overall, 2575 images of BBJ were created by using the above method, of which 2500 images were selected. Thus we had two training sets of BBJ images,

1. Training set 1 = 510 positive images and 5000 negative images.
2. Training set 2 = 2500 positive images and 5000 negative images.

5.3.2 Generating Negative Samples

Training requires positive as well as negative samples. For training different stages of the cascade, a number of negative, or background, samples are required. Basically, negative samples are the images not containing the object of interest, i.e a BBJ face. In order to train different stages, different negative samples are required. The negative samples should also represent all sorts of backgrounds. Thus to create a huge database of negative samples, patches are cropped from the images that are guaranteed not to contain a BBJ face. This was done by using Python coding. The negative database for BBJ-based training was created mostly from natural images, and Figure 5.6 shows examples of these images.

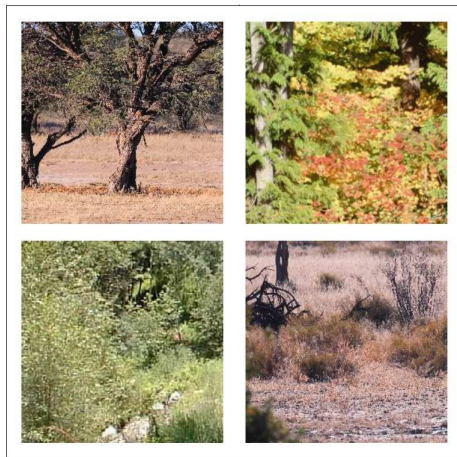


Figure 5.6: Background images for BBJ based training system.

5.4 Image Processing

Images are made up of many pixels. Each pixel provides the intensity information at that point. Different types of image provide different information to the system. Information within an image varies greatly, according to the image type, size, illumination, etc. Thus, before introducing images to the system, the images are pre-processed.

5.4.1 Greyscale Conversion

Colour images are often built from several stacked colour channels, each of which represents the value levels of a given channel. For example, RGB images are composed of three independent channels, for red, green and blue as primary colour components, CMYK images have four channels for cyan, magenta, yellow and black ink plates, etc. Prior to using images for training, Viola and Jones converted images to greyscale to reduce the information within an image that achieves a high frame rate [77]. The value of each pixel in a greyscale image is a single sample, i.e. it carries only intensity information, and pixels in an image are stored in an 8-bit integer. In this research, Python coding was used to convert colour images to greyscale. Figure 5.7 shows an original image and an image that has been converted to greyscale.



Figure 5.7: Original image and its greyscale version.

5.4.2 Image Cropping and Resizing

The images of BBJ contain a great deal of background. In order to reduce the amount of information and background in the images, the object of interest is cropped from the images. In this project, the object of interest is the BBJ face. A database of BBJ faces was created by cropping BBJ faces from all images. The BBJ face was cropped from the eyes instead of the full face, as shown in Figure 5.8, to allow the background to be discarded. Further, the cropped images were resized to $24 * 24$. The cropping of images was done by using the GIMP graphic editor, whereas the resizing was done using Python.

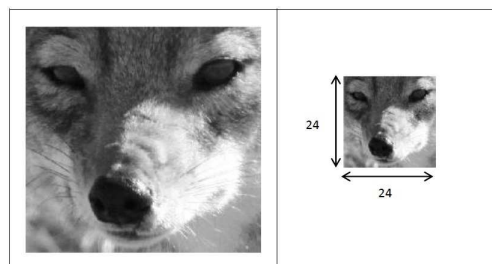


Figure 5.8: Cropped and resized image of BBJ.

5.4.3 Variance Normalisation

After all the images are cropped, resized and converted to greyscale, they are variance normalised. Variance normalisation compensates for different lighting conditions in images. The variance normalisation is explained in detail in section 3.5. The images are variance normalised by using Python. Figure 5.9 shows the original image and the variance normalised image.

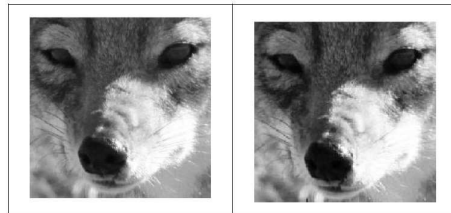


Figure 5.9: Original image and variance normalised image.

5.5 Selection of BBJ Database

As has been seen so far, two databases of BBJ faces have been created as training set 1 and training set 2. Training set 2 was created from the original 510 images by using techniques such as rotation and flipping. The reliability of the newly created database was tested by a simple experiment using OpenCV-Haartraining. The two systems were trained using the same parameters, i.e the same hit-rate, false positive rate, number of negative images and a basic set of Haar-features, but a different training set. One system was trained using training set 1 and other using training set 2. The two systems were evaluated on a testing set containing 137 images of BBJ. The ROC curve in Figure 5.10 shows the results of the two systems. Figure 5.10 shows that the two systems are fairly similar. For a higher false positive rate, training set 2 shows a higher detection than training set 1. However, as the false positive rate decreases, training set 1 provides a higher detection rate compared to training set 2. Along with a higher detection rate, the system is also expected to provide fewer false positives. For the low false alarm rate, training set 1 consistently provides the better detection rates. Thus, the training data used to perform further experiments included training set 1, i.e 510 original images of BBJ and 5000 negative (background) images.

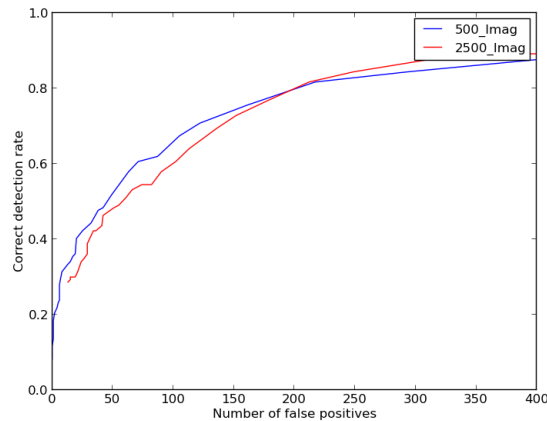


Figure 5.10: ROC curve for two different training datasets of BBJ. The two systems were evaluated on 137 images of BBJ.

5.6 Summary

The human face databases are easily available over the internet, whereas the database for BBJ needed to be created. This chapter has summarised the preparation of the database of BBJ images. The BBJ images were collected using different methods, including downloading images from web, taking photographs of BBJ, and extracting images from videos. However, of the three methods, method 1 (section 5.2.1) provided the majority of the images. Once the images had been collected, the number of BBJ images was increased by using image rotation and image flipping. Thus, two sets of BBJ images were created as training set 1 and training set 2 (section 5.3.1). Along with the positive images, training different stages of cascade also required negative images. The negative images were created by cropping patches from the images which were guaranteed not to contain the BBJ. Most of the natural images were used to create the negative database. Before introducing an image for the training, the images needed to be pre-processed. The pre-processing of the images was done by converting colour images to greyscale. The object of interest was then cropped from the images to reduce any extra information or background, and the image resized to $24 * 24$. In this project, the object of interest was the BBJ face. Finally, the images were variance normalised to compensate for different lighting conditions. Finally, the chapter ended with the comparative experiment of the two datasets, which concluded that training set 1 provided higher detection rates at lower false positive rates compared to training set 2.

Chapter 6

Jackal Face Detection

6.1 Introduction

This chapter presents the implementation of the Viola-Jones algorithm to detect BBJ faces in images. The algorithm was expected to find the BBJ faces in images if there were any, and Figure 6.1 shows examples of the images. The aim was to develop a system, using the Viola-Jones object detection algorithm, that would be able to locate BBJ faces in images. Originally the Viola-Jones object detection (VJOD) algorithm had been developed to detect a human face in images; however, the algorithm can be used to detect a variety of objects. Using this idea, an object detection system to detect BBJ faces in images was implemented. As the human face shows some distinctive features, likewise features can also be found on a BBJ face - e.g. two dark eyes separated by a light region, dark eyes with a lighter region present below the eyes, etc. Using such features, a system can be trained to find similar features in target images. The primary requirement for the implementation is the training data. The previous chapter explained how a training dataset for BBJ faces was created. On the basis of the initial experiment performed as described in section 5.5, training set 1 of BBJ images was used for further experiments. The following section explains the implementation of the BBJ detection system.



Figure 6.1: Examples of images of BBJ from test data set.

6.2 Implementation of Jackal Detection System

The jackal detection system was implemented using the Python programming language. An Asymboost boosting algorithm was used for feature selection, and the training process was accelerated by using multiprocessing to select the best Haar feature of each type. The process of developing a detection system started with introducing a set of Haar features and training images to a boosting algorithm. The implementation of the detection system included the steps below.

6.2.1 Pre-Processing of Images

The database of positive and negative images contains the cropped images. However, these images need some pre-processing before being input to the algorithm. The pre-processing of images was done using Python coding. The code for the pre-processing of images includes the following steps:

1. Open an image and resize it to $24 * 24$.
2. Convert image to greyscale.

3. Carry out variance normalisation of the image.
4. Save the variance normalised file.

6.2.2 Haar Feature File

The detection system was implemented by introducing training images and a Haar feature file to a boosting algorithm. As positive and negative samples had already been created, the aim was now to create a Haar feature file. A single Haar feature is a set of (x, y, w, h, f, p) , where x, y is the location, w, h is the size, f is the feature type and p is the parity of a Haar feature. The Haar feature file contains all Haar features of each type. The Haar feature file is created by calculating a Haar feature of types A, B, C and D at all possible locations of all possible sizes. The number of Haar features in a $24 * 24$ size window is 134736.

6.2.3 Implementation

The implementation of the BBJ detection system is explained in the following steps:

1. Write a Python code for the Asymboost boosting algorithm.
2. Allow multiprocessing for feature selection.
3. Input a Haar feature file and BBJ training images into a boosting algorithm.
4. Save the output of the algorithm into a file named 'strong classifier'.
5. Evaluate strong classifier on validation data to check the false positive rate of the system.
6. If the false positive rate is greater than the desired false positive rate, train another strong classifier using positives and a separate set of negative images that have passed through the previous stage.
7. Add strong classifiers into a cascade until the desired false positive rate is achieved.

6.3 Experiments

Using the implementation explained in the previous section, two experiments were performed, which are explained below:

6.3.1 Experiment 1

1. Purpose: Experiment 1 was a simple experiment of training a Haar cascade with a basic set of Haar features. The basic set of Haar features showed good results for human faces. The aim of this experiment was to determine to what extent the basic set of Haar features would work for the BBJ face.
2. Execution: Towards this aim, a system was trained using training set 1, i.e. 510 original images of BBJ and 5000 negative images. As the images had been resized to $24 * 24$, the total number of Haar features used was 134736. An Asymboost boosting algorithm was used for feature selection. Once a strong classifier had been trained, it was evaluated on a validation data set. The strong classifiers in a cascade were trained on the positives and false positives from the previous stage. In this experiment a cascade of 10 classifiers was trained; in other words, the detector was a cascade of 10 classifiers. The detector was scanned across the images at different scales and locations, with the starting scale of $24 * 24$, using a factor of 1.25 apart. The set of different scales used for the detector included $24 * 24$, $30 * 30$, $38 * 38$, $48 * 48$. The detector was also scanned across the location by shifting the detector by 2 pixels each time.
3. Result:

A detector was evaluated on a testing set that included 137 images of BBJ in which 147 frontal faces of BBJ were present. Figure 6.2 shows the ROC curve of a cascade of 10 strong classifiers on the testing set.
4. Interpretation of results: The ROC curve of the cascade seen in Figure 6.2 showed poor results, as there were many false positives. Again, the detection rate of the system was also up to 0.625, with 1182 false positives. The reason for this could be that either the number of BBJ images used for training or the basic set of Haar features was not sufficient. Therefore, in the next experiment, one additional type of upright Haar feature was evaluated.

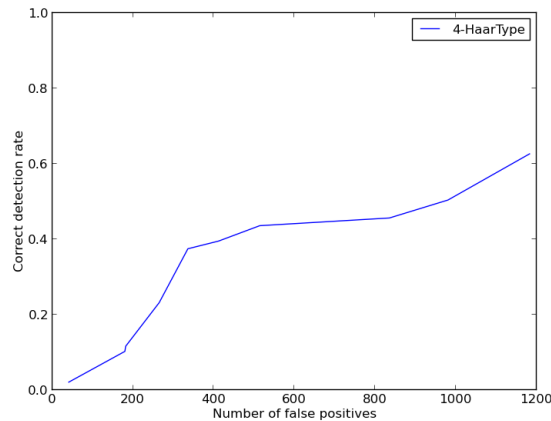


Figure 6.2: ROC curve for a 10 stage detector developed using a basic set of Haar features. The detector was evaluated on 137 images, which contained 147 frontal face images of BBJ. The total number of windows scanned by the detector was 4712388.

6.3.2 Experiment 2

As seen in Experiment 1, the basic set of Haar features was not sufficient to detect BBJ faces effectively. Therefore, it was decided to add one more Haar feature to the basic set of Haar features. This feature was used with the assumption that the spacing between human eyes was approximately one eye's width, while for a jackal it is about two eye widths. Figure 6.3 elaborates on the idea. This experiment was done using a set of five types of Haar feature including four basic Haar features and a feature described in Figure 6.4.

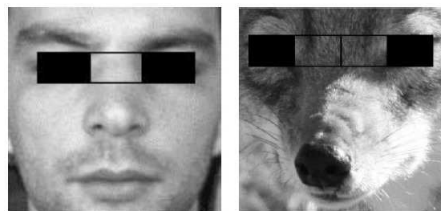


Figure 6.3: Figure elaborating the idea that the spacing between human eyes is approximately one eye's width, while for a jackal it is about two eye widths.

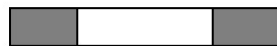


Figure 6.4: Upright Haar feature.

1. Purpose: Here the experiment was performed to determine whether the set of five types of Haar feature shows better accuracy compared to the basic set of Haar features.
2. Execution: The training data for this experiment was the same as that for preceding one, which included 510 BBJ images and 5000 negative images. The Haar features used

in this experiment was a basic set of Haar features and an upright Haar feature, as shown in Figure 6.3. An Asymboost algorithm was used for feature selection. The strong classifiers in a cascade were trained by using the positives and false positives from the previous stage. Each time a new strong classifier was added to the cascade, the cascade was evaluated on a validation data set to determine the false positive rate of the system. The strong classifiers were added to the cascade until the desired false positive rate had been achieved. In this experiment the detector was a cascade of 11 classifiers.

3. Result: The detector was evaluated on the same testing dataset, which included 137 images of BBJ. The scaling and location of the detector were kept constant as in the Experiment 1. Figure 6.5 shows the ROC curve of a cascade of 11 classifiers.

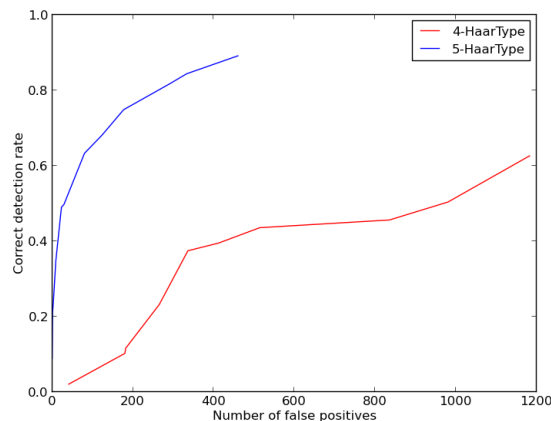


Figure 6.5: ROC curve for a 10 stage detector developed using four and five types of Haar features. The detector was evaluated on 137 images, which contained 147 frontal face images of BBJ. The total number of windows scanned by the detector was 4712388.

4. Interpretation of results: Figure 6.5 shows the comparative ROC curve of two detectors developed using four types and five types of Haar features respectively. The detector developed using five types of Haar features shows a higher detection rate with fewer false positives, whereas the detector developed using four types of Haar features had a lower detection rate with a higher false positive rate. Again it is clear that the training set used for training is not responsible for a lower detection rate in experiment 1.



Figure 6.6: An example images from the BBJ test-set. A detector of 11 classifiers is evaluated on example images by adjusting the threshold, so as to yield maximum detection and minimum false positive rate.

6.4 Result and Discussion

The detector of 11 classifiers was evaluated on examples of images by adjusting the threshold, so as to yield maximum detection and minimum false positive rates.

The final BBJ detector was a cascade of 11 classifiers. As the positive training set contained only 510 images, the detector was developed with only 11 stages, since training a larger number of stages also increases the number of false negatives. Generating a larger number of negative images is also time consuming. In addition, 5000 negative images were used for each classifier

Table 6.1: This table illustrates the description given in Figure 6.6. Each image in Figure 6.6 was of a different size, and the total number of windows scanned and number of false positives in each image are given in the table. The table also describes the number of positives actually present in the respective figures and the number of true positives detected.

Figure	Number of true positives present in image	Number of windows scanned	Number of true positives detected	Number of false positives	Number of false negatives
(a)	1	35414	1	1	0
(b)	1	35281	1	0	0
(c)	1	35303	1	1	0
(d)	2	35214	1	0	1
(e)	1	35214	1	0	0
(f)	2	35181	2	1	0
(g)	2	82987	2	1	0
(h)	1	35071	1	1	0
(i)	2	35188	1	1	1
(j)	2	35100	2	0	0
(k)	2	35141	2	2	0
(l)	1	35100	1	2	0
(m)	3	35443	3	2	0
(n)	1	35337	0	1	1
(o)	1	25105	0	2	1
(p)	2	48576	2	1	0
(q)	1	35197	1	3	0
(r)	1	35270	1	3	0
(s)	1	48177	0	1	1
(t)	1	69457	1	1	0

in a cascade, so as to discard the highest possible number of negatives in the early stages of the cascade, to achieve the expected false positive rate. In order to broaden and elaborate on the results of the detection system, 20 images were randomly selected from the test set so as to find the number of false positives and false negatives in each image. The threshold of the final stage

of a cascade was adjusted in such a way as to achieve maximum detection with a minimum of false positives. With the adjusted threshold, the system was evaluated on the 20 images. Post-processing was done on the images to merge the overlapping as discussed in section 3.7.

Figure 6.6 shows the results obtained by the detector. Table 6.1 illustrates the results from 20 images. In the 20 images, 29 frontal BBJ faces were present, of which 25 were detected by the detector. As the size of each image was different, the total number of windows scanned for each image was different. The column headed ‘Number of windows scanned’ shows the number of windows scanned for each image. The total number of windows scanned was 802756, among which there were 24 false positives.

The detection rate of the system can be determined by evaluating the detector on a full test set. The final test set contained 137 images in which there were 147 frontal faces of BBJ. Using this threshold, the system detected 114 faces of BBJ, with 224 false positives. The detection rate of the system can be calculated as:

$$\text{Detection rate} = \frac{\text{Total number of positives detected}}{\text{Total number of true positives present in images}} \quad (6.1)$$

The detector detected 114 faces out of 147. Thus, the detector achieved 78% detection with a false positive rate of 0.00623%. The detection rate of the system can be increased by lowering the threshold of the final stage of the system. However, as the threshold of the system decreases, the false positive rate of the system also increases. This number of false negatives can be expected, since the starting scale of this detector is $24 * 24$. If the starting scale were to be decreased, the false negative rate could be decreased, but this would also increase the false positives.

Finally, it can be concluded that this 11 stage detector with five types of Haar feature performs fairly well in the detection of BBJ faces. On the other hand, to improve performance, the false positive rate of the system could be decreased by adding a few stages to the cascade.

6.5 Summary

This chapter has presented the implementation of a jackal detection system. The detection system is implemented using training set 1 on the basis of an experiment performed as described

in section 5.5. An Asymboost algorithm was used for feature selection. The pre-processing of images and the implementation of the system was done using Python. A system was implemented according to the implementation process explained in detail in section 6.2, with a basic set of Haar features (section 6.3.1). However, it was found that the system developed using the basic set of Haar features did not yield a good detection rate. Thus, in Experiment 2 (section 6.3.2), one more upright Haar feature was added to the basic set, with the assumption that the relative spacing between human eyes is about one eye's width, while for jackal it is about two eye widths. It was found that the system trained in Experiment 2 achieved a higher detection rate when compared with the system developed in Experiment 1. Finally, the chapter is concluded with a summary of the results and a discussion of the final detector.

Chapter 7

Conclusion and Future Work

7.1 Conclusion

In this project a system to detect BBJ faces in images was implemented. The system was based on the Viola-Jones object detection algorithm, which uses Haar-like features together with the Adaboost classification algorithm. The development of the object detection algorithm was basically motivated by the human face detection problem. Thus, in addition to BBJ detection system, a human face detection system was also developed by the researcher.

The human face detection system was implemented using the Asymboost classification algorithm instead of the Adaboost boosting algorithm, since the Adaboost algorithm reduced classification errors instead of reducing the number of false negatives. The comparative ROC curve of the two systems, Adaboost and Asymboost, clearly shows that the Asymboost algorithm works better than the Adaboost one. Therefore, the final face detector was implemented using the Asymboost algorithm and the basic set of Haar features. The detector achieved a detection rate of 0.88 with a false positive rate of 0.00001 on the BioID-face database. However, the detection rate of the system could be increased by reducing the threshold of the final stage, but this would also increase the number of false positives. For the human face detection system, we conclude that the 16 stage human face detector performs well for the specific resolution images and achieves a good detection rate when there are individual faces in images or when the faces in images are well separated.

The BBJ detection system was implemented using an Asymboost algorithm and a basic set of Haar features. However, it was found that with the basic set of Haar features, the detection

rate of the system was lower than expected. Therefore, one more upright Haar-feature was added to the basic set of Haar-features with the assumption that the distance between the two eyes of the jackal is approximately two eye widths. The final BBJ detection system was trained using five types of Haar feature. The final BBJ detector achieved a detection rate of 0.776. The detection rate of the BBJ detector is lower than that of the human face detector, but this was to be expected as the BBJ detection system was trained using only 510 positive images. Again, the positive training images for human faces were captured under the desired conditions, whereas the positive training images for BBJ are natural and no special alignment was done.

7.2 Future Work

As this project was concentrated mainly on the detection of BBJ, the future work on this project will comprise mainly BBJ detection. The lessons learned throughout the implementation suggest that the system can be improved by attention to the following points:

1. Implementing a detection system using all types of upright and tilted Haar features
2. Increasing the number of positive training images

Although the results of BBJ detection system can be improved on the above mentioned points, it is still worth noting that the current 11 stage detector performs fairly satisfactorily.

Bibliography

- [1] G. Yang and T. S. Huang, “Human face detection in a complex background,” *Pattern recognition*, vol. 27, no. 1, pp. 53–63, 1994.
- [2] A. Liaw and M. Wiener, “Classification and regression by randomforest,” *R news*, vol. 2, no. 3, pp. 18–22, 2002.
- [3] N. Natrass and B. Conradie, “Jackal narratives and predator control in the Karoo, South Africa,” 2013.
- [4] L. Breiman, “Random forests,” *Machine learning*, vol. 45, no. 1, pp. 5–32, 2001.
- [5] Y. Amit and D. Geman, “Shape quantization and recognition with randomized trees,” *Neural computation*, vol. 9, no. 7, pp. 1545–1588, 1997.
- [6] A. Goldblatt, “Agriculture: Facts and trends.” http://awsassets.wwf.org.za/downloads/facts_b
Accessed: October 2013.
- [7] Anonymous, “A profile of the South African mutton market value chain,” 2011.
Department: Agriculture, Forestry and Fisheries, South Africa.
- [8] S. David, *Analysis and quantification of the South African red meat value chain*. PhD thesis, 2011.
- [9] D. Rowe-Rowe, “Predation by black-backed jackals in a sheep-farming region of Natal,” *Journal of the Southern African Wildlife Management Association*, vol. 5, pp. 79–81, 1975.
- [10] D. Lawson, “The effects of predators on sheep farming in Natal-an opinion survey,” *South African Journal of Wildlife Research*, vol. 19, no. 1, pp. 4–10, 1989.
- [11] Anonymous, “Can big bucks solve the predator problem?,” *Farmer’s weekly*, July 18, 2011.

- [12] C. J. Skead, A. Boshoff, G. I. H. Kerley, and P. Lloyd, *Historical incidence of the larger land mammals in the broader northern and western Cape*. Centre for African Conservation Ecology, Nelson Mandela Metropolitan University, 2011.
- [13] U. d. V. Pienaar, "Predator-prey relationships amongst the larger mammals of the Kruger National Park," *Koedoe*, vol. 12, pp. 108–176, 1969.
- [14] A. J. Loveridge and J. A. Nel, *Black-backed jackal*, ch. 6.3. IUCN/SSC Canid Specialist Group. IUCN, Gland, Switzerland and Cambridge, United Kingdom, 2004.
- [15] J. C. Ray, L. Hunter, and J. Zigouris, *Setting conservation and research priorities for larger African carnivores*, vol. 24. Wildlife Conservation Society New York, 2005.
- [16] W. Beinart, "The night of jackal: sheep, pastures and predators in the Cape," *Past and Present*, no. 158, pp. 172–206, 2011.
- [17] W. Beinart, *The Rise of Conservation in South Africa: settlers, livestock, and the environment 1770-1950*. Oxford University Press, 2008.
- [18] Anonymous, "Animal damage control institue." <http://www.jackal.co.za/>. Accessed: October 2013.
- [19] "Mountain lion foundation." <http://www.mountainlion.org/portalprotectguardanimals.asp>. Accessed: August 2013.
- [20] H. Anna, "The trapping truth." <http://www.landmarkfoundation.org.za/gin-traps.html>. Accessed: September 2013.
- [21] S. Thorpe, D. Fize, C. Marlot, *et al.*, "Speed of processing in the human visual system," *Nature*, vol. 381, no. 6582, pp. 520–522, 1996.
- [22] T. Burghardt and J. Calic, "Real-time face detection and tracking of animals," in *Neural Network Applications in Electrical Engineering, 2006. NEUREL 2006. 8th Seminar on*, pp. 27–32, IEEE.
- [23] C. Peijiang, "Moving object detection based on background extraction," in *Computer Network and Multimedia Technology, 2009. CNMT 2009. International Symposium on*, pp. 1–4, IEEE.

- [24] J. C. Nascimento and J. S. Marques, "Performance evaluation of object detection algorithms for video surveillance," *Multimedia, IEEE Transactions on*, vol. 8, no. 4, pp. 761–774, 2006.
- [25] B. T. Koik and H. Ibrahim, "A literature survey on animal detection methods in digital images," *International Journal of Future Computer and Communication*, Vol. 1, No. 1, June 2012.
- [26] M. S. Zahrani, K. Ragab, and A. U. Haque, "Design of gps-based system to avoid camel-vehicle collisions: A," *Asian Journal of Applied Sciences*, vol. 4, no. 4, pp. 362–377, 2011.
- [27] Y. Oishi and T. Matsunaga, "Automatic detection of moving wild animals in airborne remote sensing images," in *Geoscience and Remote Sensing Symposium (IGARSS), 2010 IEEE International*, pp. 517–519, IEEE, 2010.
- [28] D. Tahmoush and J. Silvius, "Modeled gait variations in human micro-doppler," in *Radar Symposium (IRS), 2010 11th International*, pp. 1–4, IEEE, 2010.
- [29] S.-H. Kim, D.-H. Kim, and H.-D. Park, "Animal situation tracking service using rfid, gps, and sensors," in *Computer and Network Technology (ICCNT), 2010 Second International Conference on*, pp. 153–156, IEEE, 2010.
- [30] J. S. Ting, S. Kwok, W. Lee, A. H. Tsang, and B. C. Cheung, "A dynamic rfid-based mobile monitoring system in animal care management over a wireless network," in *Wireless Communications, Networking and Mobile Computing, 2007. WiCom 2007. International Conference on*, pp. 2085–2088, IEEE, 2007.
- [31] M. Fabre-Thorpe, A. Delorme, C. Marlot, and S. Thorpe, "A limit to the speed of processing in ultra-rapid visual categorization of novel natural scenes," *Journal of cognitive neuroscience*, vol. 13, no. 2, pp. 171–180, 2001.
- [32] A. Torralba and A. Oliva, "Statistics of natural image categories," *Network: computation in neural systems*, vol. 14, no. 3, pp. 391–412, 2003.

- [33] F. A. Wichmann, J. Drewes, P. Rosas, and K. R. Gegenfurtner, “Animal detection in natural scenes: Critical features revisited,” *Journal of Vision*, vol. 10, no. 4, 2010.
- [34] C. Peijiang, “Moving object detection based on background extraction,” in *Computer Network and Multimedia Technology, 2009. CNMT 2009. International Symposium on*, pp. 1–4, IEEE, 2009.
- [35] J. C. Nascimento and J. S. Marques, “Performance evaluation of object detection algorithms for video surveillance,” *IEEE Transactions on Multimedia*, vol. 8, no. 4, pp. 761–774, 2006.
- [36] T. Burghardt and J. Calic, “Real-time face detection and tracking of animals,” in *Neural Network Applications in Electrical Engineering, 2006. NEUREL 2006. 8th Seminar on*, pp. 27–32, IEEE, 2006.
- [37] V. Mitra, C.-J. Wang, and S. Banerjee, “LIDAR detection of underwater objects using a neuro-svm-based architecture,” *Neural Networks, IEEE Transactions on*, vol. 17, no. 3, pp. 717–731, 2006.
- [38] M. Parikh and M. Patel, “Animal detection using template matching algorithm.” MBICT, Gujarat Technological University, India.
- [39] H. A. Rowley, S. Baluja, and T. Kanade, “Neural network-based face detection,” *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 20, no. 1, pp. 23–38, 1998.
- [40] H. Schneiderman and T. Kanade, “A statistical method for 3d object detection applied to faces and cars,” in *Proceedings. IEEE Conference on Computer Vision and Pattern Recognition, 2000.*, vol. 1, pp. 746–751, IEEE.
- [41] P. Viola and M. J. Jones, “Robust real-time face detection,” *International Journal of Computer vision*, vol. 57, no. 2, pp. 137–154, 2004.
- [42] Anonymous, “Face detection - wikipedia, the free encyclopedia.” http://en.wikipedia.org/wiki/Face_detection.
- [43] P. F. da Carrera, “Face recognition algorithms,” June, 2010.

- [44] M.-H. Yang, D. J. Kriegman, and N. Ahuja, “Detecting faces in images: A survey,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 24, no. 1, pp. 34–58, 2002.
- [45] S. A. Sirohey, “Human face segmentation and identification,” 1998.
- [46] D. Chetverikov and A. Lerch, “Multiresolution face detection,” *Theoretical Foundations of Computer Vision*, vol. 69, pp. 131–140, 1993.
- [47] J. Yang and A. Waibel, “A real-time face tracker,” in *Applications of Computer Vision, 1996. WACV’96., Proceedings 3rd IEEE Workshop on*, pp. 142–147, IEEE.
- [48] M. F. Augusteijn and T. L. Skufca, “Identification of human faces through texture-based feature recognition and neural network technology,” in *IEEE International Conference on Neural Networks, 1993.*, pp. 392–398, IEEE.
- [49] H. P. Graf, T. Chen, E. Petajan, and E. Cosatto, “Locating faces and facial parts,” in *Proc. First Intl Workshop Automatic Face and Gesture Recognition*, pp. 41–46.
- [50] H. P. Graf, E. Cosatto, D. Gibbon, M. Kocheisen, and E. Petajan, “Multi-modal system for locating heads and faces,” in *Proceedings of the Second International Conference on Automatic Face and Gesture Recognition, 1996.*, pp. 88–93, IEEE.
- [51] J.-C. Terrillon, M. David, and S. Akamatsu, “Detection of human faces in complex scene images by use of a skin color model and of invariant fourier-mellin moments,” in *Proceedings. Fourteenth International Conference on Pattern Recognition, 1998.*, vol. 2, pp. 1350–1355, IEEE.
- [52] J.-C. Terrillon, M. David, and S. Akamatsu, “Automatic detection of human faces in natural scene images by use of a skin color model and of invariant moments,” in *Proceedings. Third IEEE International Conference on Automatic Face and Gesture Recognition, 1998.*, pp. 112–117, IEEE.
- [53] I. Craw, H. Ellis, and J. R. Lishman, “Automatic extraction of face-features,” *Pattern Recognition Letters*, vol. 5, no. 2, pp. 183–187, 1987.
- [54] A. Lanitis, C. J. Taylor, and T. F. Cootes, “Automatic face identification system using flexible appearance models,” *Image and vision computing*, vol. 13, no. 5, pp. 393–401, 1995.

- [55] N. da Vitoria Lobo and Y. H. Kwon, “Face detection using templates,” 1998.
- [56] M. Kass, A. Witkin, and D. Terzopoulos, “Snakes: Active contour models,” *International Journal of Computer Vision*, vol. 1, no. 4, pp. 321–331, 1988.
- [57] K.-M. Lam and H. Yan, “Fast algorithm for locating head boundaries,” *Journal of Electronic Imaging*, vol. 3, no. 4, pp. 351–359, 1994.
- [58] F. Leymarie and M. D. Levine, “Tracking deformable objects in the plane using an active contour model,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 15, no. 6, pp. 617–634, 1993.
- [59] A. L. Yuille, P. W. Hallinan, and D. S. Cohen, “Feature extraction from faces using deformable templates,” *International Journal of Computer Vision*, vol. 8, no. 2, pp. 99–111, 1992.
- [60] B. Scassellati, “Eye finding via face detection for a foveated active vision system,” in *AAAI/IAAI*, pp. 969–976.
- [61] “Support vector machine.” http://en.wikipedia.org/wiki/Support_vector_machine. Accessed: October 2013.
- [62] K. Pearson, “Liii. on lines and planes of closest fit to systems of points in space,” *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, vol. 2, no. 11, pp. 559–572, 1901.
- [63] L. Sirovich and M. Kirby, “Low-dimensional procedure for the characterization of human faces,” *JOSA A*, vol. 4, no. 3, pp. 519–524, 1987.
- [64] M. Turk and A. Pentland, “Eigenfaces for recognition,” *Journal of cognitive neuroscience*, vol. 3, no. 1, pp. 71–86, 1991.
- [65] Y. H. Ann, *Real-time face tracking*. PhD thesis, National University of Singapore, 2005/2006.
- [66] T. K. Ho, “The random subspace method for constructing decision forests,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 20, no. 8, pp. 832–844, 1998.

- [67] A. Ferreira, “Survey on boosting algorithms for supervised and semi-supervised learning,” *Institute of Telecommunications*, 2007.
- [68] Y. Freund and R. E. Schapire, “Experiments with a new boosting algorithm,” in *ICML*, vol. 96, pp. 148–156.
- [69] J. Friedman, T. Hastie, and R. Tibshirani, “Additive logistic regression: A statistical view of boosting (with discussion and a rejoinder by the authors),” *The annals of statistics*, vol. 28, no. 2, pp. 337–407, 2000.
- [70] R. E. Schapire and Y. Singer, “Improved boosting algorithms using confidence-rated predictions,” *Machine learning*, vol. 37, no. 3, pp. 297–336, 1999.
- [71] S. Z. Li and A. K. Jain, *Handbook of face recognition*. springer, 2011.
- [72] S. Z. Li and Z. Zhang, “Floatboost learning and statistical face detection,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 26, no. 9, pp. 1112–1123, 2004.
- [73] C. Liu and H.-Y. Shum, “Kullback-Leibler boosting,” in *Proceedings. 2003 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2003.*, vol. 1, pp. I-587–I-594 vol. 1, IEEE.
- [74] O. H. Jensen, *Implementing the Viola-Jones face detection algorithm*. PhD thesis, Technical University of Denmark, DTU, DK-2800 Kgs. Lyngby, Denmark, 2008.
- [75] M. Jones and P. Viola, “Fast multi-view face detection,” *Mitsubishi Electric Research Lab TR-20003-96*, vol. 3, p. 14, 2003.
- [76] C. P. Papageorgiou, M. Oren, and T. Poggio, “A general framework for object detection,” in *Sixth International Conference on Computer Vision, 1998.*, pp. 555–562, IEEE.
- [77] P. Viola and M. J. Jones, “Robust real-time face detection,” *International journal of computer vision*, vol. 57, no. 2, pp. 137–154, 2004.
- [78] “Imagemagick identify command-line tool.” <http://www.imagemagick.org/script/identify.php>. Accessed: October 2014.

- [79] “Tutorial: Opencv haartraining (rapid object detection with a cascade of boosted classifiers based on haar-like features).” <http://note.sonots.com/SciSoftware/haartraining.html#wf43989b>. Accessed: October 2014.
- [80] “OPENCV HAAR-TRAINING.” <http://abhishek4273.com/2014/02/10/opencv-haar-training/>. Accessed: October 2014.

Appendix A

OpenCV Haartraining

An OpenCV-Haartraining allows one to detect human faces and eyes. A similar cascade can be trained using OpenCV to detect various other objects. An OpenCV-Haartraining is divided into three steps: the preparation of data, training, and testing. This process is described in the following section.

A.1 Preparation of Data

For the purposes of training, thousands of images of the object of interest are needed, along with background images. Images containing the object of interest are termed positive images, whereas background images are termed negative images. The positive images can be collected by downloading images of the object of interest from the web or by taking photographs. Negative images may be any images that do not contain the object of interest. The OpenCV-Haartraining needs description files for both the positive and the negative images. Generally an image contains more than one object. A description file of positive images contains the description of the objects in an image. A description file of positive images can be created using the ‘identify’ command of Imagemagick [78]. The command for creating a description file is [79, 80]:

```
$ find [dir name] -name '*. <jpg >' -exec identify -format '%i
1 0 0 %w %h'\{\} \; > Postive.dat
```

The format of a description file should be:

```
[filename] [Number of Objects] [x y w h]
```

where x, y are the x , and y coordinates, w is the width and h is the height of the bounding box of an object. The description file of cropped positive images looks like this:

```
/Project/pos/p1.jpg 1 0 0 114 123
/Project/pos/p2.jpg 1 0 0 70 53
/Project/pos/p3.jpg 1 0 0 210 234
```

However, all the positive images in the description file needs to be packed in a `.vec` file. The `vec` file is created using the ‘createsamples’ utility provided by OpenCV. The command to pack positive images into the `vec` file is [79, 80],

```
$ OpenCV_createsamples -info Positive.dat -vec Positive.vec -w 24
-h 24
```

where `Positive.dat` is the description file of positive images, and `Positive.vec` is the `vec`file of positive images in which positive images are resized to a given width (`-w`) and height (`-h`) and packed as thumbnails.

The description file of negative images contains only the filenames of the negative images and can thus be called a collection file. The collection file of negative images is created using the following command:

```
$ find [image dir.] -name '*.jpg' > Negative.dat
```

The collection file of negative images looks like this:

```
/Project/neg/n1.jpg
/Project/neg/n2.jpg
/Project/neg/n3.jpg
```

A.2 Training

OpenCV-Haartraining generates an xml cascade file that can be used to detect an object. To generate this xml file, we need a vecfile of positive images and a collection file of negative images. The different parameters required to generate a cascade include: number of stages, minimum hitrate, and maximum false alarm rate. The OpenCV-Haartraining is initialised using the command [79],

```
$ OpenCV-haartraining -data Haar -vec Positive.vec -bg  
Negative.dat -nstages 15 -minhitrate 0.999 -maxfalsealarm  
0.5 -npos 2300 -nneg 4548 -w 24 -h 24 -mem 1000 -mode ALL
```

```
-data: folder where haartraining will be saved  
-vec: vec file of positive images  
-bg: collection file of negative images  
-nstages: number of stages in a cascade  
-minhitrate: minimum hit rate of single stage classifier  
-maxfalsealarm: false alarm rate of single stage classifier  
-npos: number of positive images  
-nneg: number of negative images  
-w: width of image  
-h: height of image  
-mem: memory allotted to the haartraining  
-mode: BASIC / CORE / ALL
```

Haartraining is completed when the required false alarm rate is achieved. Sometimes, if the required false alarm rate has been achieved, a cascade will contain fewer stages than stated in the command. The training generates both the xml file and the folder containing Haartraining, which can be used for testing the cascade.

A.3 Testing

The performance of a generated cascade can be tested using the following command:

```
$ OpenCV-performance -data Haar -w 24 -h 24 -info tests.dat
```

```
-data: folder where haartraining is saved
```

```
-w: width of image
```

```
-h: height of image
```

```
-info: collection file of testing data
```

The output of the performance utility shows the number of detections, the number of false negatives, and the number of false positives in each image. Furthermore, it also provides the table for the ROC plot.