ENERGY EFFICIENT ENABLING TECHNOLOGIES FOR SEMANTIC VIDEO PROCESSING ON MOBILE DEVICES

by

Daniel Larkin, B.Eng.

Submitted in partial fulfilment of the requirements

for the Degree of Doctor of Philosophy

Dublin City University

School of Electronic Engineering

Supervisor: Prof. Noel E. O'Connor

September, 2008

I hereby certify that this material, which I now submit for assessment on the programme of study leading to the award of Doctor of Philosophy is entirely my own work, that I have exercised reasonable care to ensure that the work is original, and does not to the best of my knowledge breach any law of copyright, and has not been taken from the work of others save and to the extent that such work has been cited and acknowledged within the text of my work.

Signed: _____
Daniel Larkin (Candidate)

ID: _____

Date: _____

Table of Contents

List of Figures

List of Tables

ABSTRACT

Semantic object-based processing will play an increasingly important role in future multimedia systems due to the ubiquity of digital multimedia capture/playback technologies and increasing storage capacity. Although the object based paradigm has many undeniable benefits, numerous technical challenges remain before the applications becomes pervasive, particularly on computational constrained mobile devices. A fundamental issue is the ill-posed problem of semantic object segmentation. Furthermore, on battery powered mobile computing devices, the additional algorithmic complexity of semantic object based processing compared to conventional video processing is highly undesirable both from a real-time operation and battery life perspective. This thesis attempts to tackle these issues by firstly constraining the solution space and focusing on the human face as a primary semantic concept of use to users of mobile devices. A novel face detection algorithm is proposed, which from the outset was designed to be amenable to be offloaded from the host microprocessor to dedicated hardware, thereby providing real-time performance and reducing power consumption. The algorithm uses an Artificial Neural Network (ANN), whose topology and weights are evolved via a genetic algorithm (GA). The computational burden of the ANN evaluation is offloaded to a dedicated hardware accelerator, which is capable of processing any evolved network topology. Efficient arithmetic circuitry, which leverages modified Booth recoding, column compressors and carry save adders, is adopted throughout the design. To tackle the increased computational costs associated with object tracking or object based shape encoding, a novel energy efficient binary motion estimation architecture is proposed. Energy is reduced in the proposed motion estimation architecture by minimising the redundant operations inherent in the binary data. Both architectures are shown to compare favourable with the relevant prior art.

List of Acronyms

1G First Generation

2G Second Generation

3G Third Generation

4G Fourth Generation

AI Artificial Intelligence

ANN Artificial Neural Network

BAB Binary Alpha Block

BAP Binary Alpha Plane

BME Binary Motion Estimation

CGM Constrained Generative Model

CIF Common Intermediate Format

CISC Complex Instruction Set Computer

CRT Cathode Ray Tube

DCT Discrete Cosine Transform

DPCM Differential Pulse Code Modulation

DWT Discrete Wavelet Transform

EA Evolutionary Algorithm

EANN Evolutionary Artificial Neural Network

FPGA Field Programmable Gate Array

FPS Frames Per Second

FSM Finite State Machine

GA Genetic Algorithm

GPU Graphics Processing Unit

GSM Global System for Mobile

HDTV High Definition Television

HMM Hidden Markov Model

HSI Hue Satursation Intensity

IDCT Inverse Discrete Cosine Transform

Kbps Kilobits per second

KLT Karhunen-Loeve Transform

LCD Liquid Crystal Display

LUT Look Up Table

Mbps Megabits per second

ME Motion Estimation

MLP Multi-Layered Perceptron

MVPS Motion Vector Predictor for Shape

MVS Motion Vector for Shape

NEAT Neuro Evolution of Augmenting Topologies

PCA Prinicipal Component Analysis

PDA Personal Digital Assistant

PE Processing Element

QCIF Quarter Common Intermediate Format

RGB Red, Green and Blue

RISC Reduced Instruction Set Computer

ROC Receiver Operator Characteristics

ROM Read Only Memory

RSST Recursive Shortest Spanning Tree

SAD Sum of Absolute Differences

SA-DCT Shape Adaptive Discrete Cosine Transform

SA-IDCT Shape Adaptive Inverse Discrete Cosine Transform

SDTV Standard Definition Television

SMS Short Messaging Service

SNoW Sparse Network of Winnows

SoC System on a Chip

SOM Self Organising Maps

SVD Singular Value Decomposition

SVM Support Vector Machine

TWEANN Topology and Weight Evolving Artificial Neural Networks

VLC Variable length Code

VoIP Voice over Internet Protocol

VOP Video Object Plane

WAP Wireless Application Protocol

List of Peer-Reviewed Publications

- Patents

    1. "Early Exit Techniques for Digital Video Motion Estimation", filed 2^{nd} May 2005, Application number PCT/IE2005/000023.

- Journals

    1. Andrew Kinane, **Daniel Larkin** and Noel O'Connor,"Energy-Efficient Acceleration of an MPEG-4 Codec", EURASIP Journal on Embedded Systems, January 2007.

- Conferences

    1. **Daniel Larkin**, Andrew Kinane and Noel O'Connor, "Towards Hardware Acceleration of Neuroevolution for Multimedia Processing Applications on Mobile Devices", in the proceedings of the International Conference on Neural Information Processing, Hong Kong, China. October 3rd–6th 2006.
    2. **Daniel Larkin**, Andrew Kinane and Noel O'Connor, "An Efficient Hardware Architecture for a Neural Network Activation Function Generator", in the proceedings of the International Symposium on Neural Networks, Chengdu, P.R. China, May 29th–31st, 2006.
    3. **Daniel Larkin**, Valentin Muresan and Noel O'Connor, "A Low Complexity Hardware Architecture for Motion Estimation", in the proceedings of the IEEE International Symposium on Circuits and Systems, Kos, Greece, 21st–24th May 2006.

4. **Daniel Larkin**, Valentin Muresan and Noel O'Connor, "An Efficient Motion Estimation Hardware Architecture for MPEG-4 Binary Shape Coding", in the proceedings of the Irish Signals and Systems Conference, Dublin, Ireland, 1st–2nd September 2005.
5. **Daniel Larkin**, Andrew Kinane, Valentin Muresan and Noel O'Connor, "Efficient Hardware Architectures for MPEG-4 Core Profile", in the proceedings of the Irish Machine Vision and Image Processing Conference, Belfast, Northern Ireland, August 30th–31st 2005.
6. Noel O'Connor, Valentin Muresan, Andrew Kinane, **Daniel Larkin**, Sean Marlow and Noel Murphy, "Hardware Acceleration Architectures for MPEG-Based Mobile Video Platforms: A Brief Overview", in the proceedings of the 4th Workshop on Image Analysis for Multimedia Interactive Services (WIAMIS), London, England, 9th–11th April 2003.

- Contributions to MPEG-4 (Non peer reviewed)

1. **Daniel Larkin**, Andrew Kinane and Noel O'Connor, "Conformance testing of Binary Motion Estimation for MPEG-4 Binary Shape Coding", Contribution 13182 to AHG on MPEG-4 Part-9: Reference Hardware ISO/IEC JTC1/SC29/WG11, Montreux, Switzerland, April 2006.
2. **Daniel Larkin**, Valentine Muresan and Noel O'Connor, "Updated Status and documentation of the Shape Coding Binary Motion Estimation Hardware Acceleration Module", Contribution N6759 to AHG on MPEG-4 Part-9: Reference Hardware ISO/IEC JTC1/SC29/WG11, Palma de Mallorca, Spain, October 2004.
3. **Daniel Larkin**, Valentine Muresan and Noel O'Connor, "Hardware Acceleration Module for MPEG-4 binary shape coding motion estimation", Contribution M11092 to AHG on MPEG-4 Part-9: Reference Hardware ISO/IEC JTC1/SC29/WG11, Redmond, USA, July 2004.

ACKNOWLEDGEMENTS

It is a pleasure to thank the many people who have contributed in making this thesis possible. Firstly, I would like to thank my supervisor, Prof. Noel O'Connor, for his guidance and support during this research. In particular, I'm very grateful for his advice and invaluable suggestions throughout the thesis writing period. I'd also like to thank all the past and present members of the hardware group, particularly Andrew Kinane, Kealan McCusker and Val Muresan, for all their technical assistance. I owe a great debt of gratitude to my mother and sisters for their support and encouragement over the years. I am also tempted to individually thank all of my friends. However, there are too many to mention, so I will simply say thank you very much to you all. Lastly, and most importantly, I wish to thank Fiona for putting up with the late hours and shortened weekends. Without her constant encouragement and love, this research would never have been completed.

CHAPTER 1

Introduction

Relentless progress in semiconductor technology is a key enabling factor in the continuous evolution of modern connected computing solutions [1][2]. Both now and historically, this evolution is driven by the diverse needs of users in academic, military, business and home environments. For example the emergence of key computing platforms (mainframe, PC, etc.) and different application families (databases, spreadsheets, data networking, computer gaming, etc) can be directly attributed to these diverse needs. Moreover, whereas once computer technology was only considered as a tool to assist in scientific and business calculations, it is now also deeply ingrained in many fundamental facets of our daily lives, such as entertainment (digital films, digital TV, etc.) and communications (Voice over Internet Protocol (VoIP), email, instant messaging, etc). With such ubiquity, the choice of which platform (server, PC, laptop, mobile device etc.) a user employs is typically a trade off between cost and/or requirements (computational resources, networking resources, interface, mobility etc). Increasingly, mobile devices such as mobile phones, smartphones, Personal Digital Assistants (PDA) etc are becoming a viable and popular platform for a range of applications. In 2005 alone there were over 800 million mobile phones sold [3][4]. This compares to approximately 200 million desktops, notebooks and x86 servers which were shipped in the same year [5]. In 2006 there were over 1 billion mobile phones sold [6].

There are many contributing factors to the popularity and success of such mobile devices. A fundamental desire for untethered voice communications fuelled the success of the initial First Generation (1G) and Second Generation (2G) mobile phones. Emerging Third Generation (3G)

mobile devices have a much broader appeal due to improved computational and telecommunications capabilities. These mobile devices allow multiple forms of effective and immediate communication. In addition, the improving general purpose computing performance of the device is enabling application convergence (e.g. telephone, video conferencing, Short Messaging Service (SMS), email, internet browsing, gaming, digital camera, digital audio/video player, mobile TV, etc) on the devices [7][8][9]. These additional applications are beneficial in both personal and corporate use scenarios, and allow "dead time" (e.g. commuting, queueing, etc) to be used more productively. This is a particularly valuable feature for todays society, which places such a high value on the commodity of time. By augmenting the technical capabilities and improving mass market appeal, mobile devices are now frequently designed and marketed as highly desirable, premium priced style icons [10][11]. It is not surprising then to find users who feel a strong sense of attachment to their mobile device. The growing market for device personalisation (ring tones, wallpapers, coloured fascias), which was worth over $3 billion worldwide in 2005, is strong evidence of the desire of a subset of users to extend their personality and individualism to their mobile device [12]. This is a clear indication of how deeply ingrained mobile devices have become in todays social fabric. Coupled with the technical advances, this demonstrates how increasingly important the ubiquitous mobile device will become as a general purpose computing platform in the future.

1.1 The Emergence of Mobile Multimedia

The widespread availability of low cost digital capture and playback devices with ever increasing storage capacity, coupled with rapid advances in signal processing, have now made digital multimedia universal [13][14][11]. These technological advances alongside changing human behaviour have affected all aspects of how multimedia is now created, stored, distributed and consumed. For example, traditional sources of video content (TV, film etc.) are being complemented by user created content (e.g. mobile-video calls, video messaging, video blogs, etc.) [15]. In this new paradigm, users are frequently empowered by the ubiquitous battery powered mobile device and motivated by factors such as a desire for graphically enriched communications, or for filling "dead time" by consuming multimedia content. These devices have the potential to offer an engaging mobile multimedia experience for the user. Furthermore, reflecting a symbiotic relationship, increased multimedia processing capabilities is considered vital for future mobile devices [16].

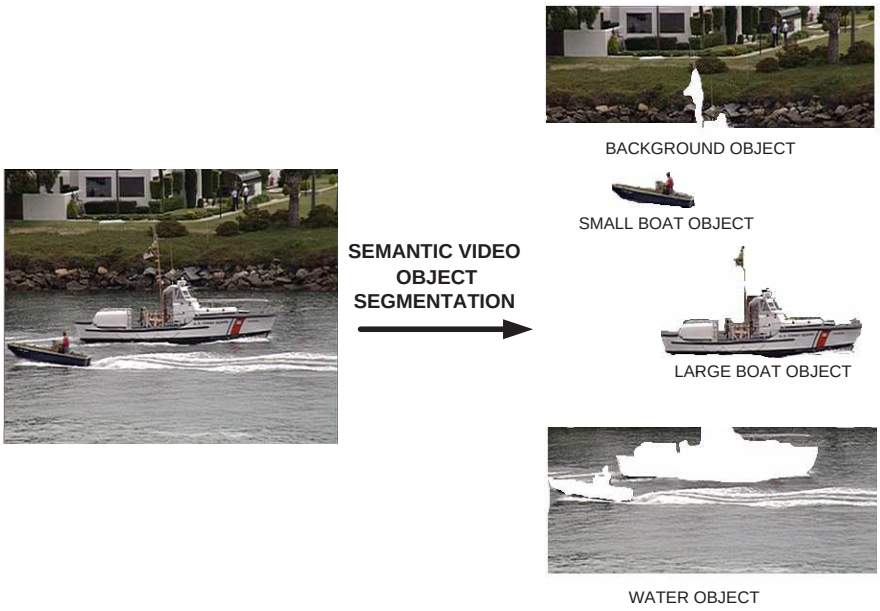


Figure 1.1: Video scene decomposed into constituent semantic objects

However, numerous technical challenges remain to be solved before next generation mobile multimedia can become a reality.

1.2 Grand Challenges Facing Next Generation Mobile Multimedia

The phenomenal growth in digital multimedia in conjunction with the emergence of convergent mobile devices has highlighted a number of open research challenges facing next generation mobile multimedia. Amongst others these challenges include semantic multimedia processing, multimedia content delivery and the issues arising as a result of the constrained resources available on a mobile device. This section briefly outlines these challenges in order to give context for the research undertaken in this thesis. A more thorough discussion of these issues is presented in Chapter 2.

1.2.1 Semantic Multimedia Processing

Rapidly increasing volumes of digital multimedia content emphasises a need for improved content structure awareness in multimedia processing algorithms. For example, this requirement becomes apparent by witnessing the difficulties encountered whilst searching large personal digital image or video libraries devoid of meta-data. Moreover, bridging the gap between a correlated collection of pixels and the overall semantics of these pixels, is of considerable benefit to the entire end to

Table 1.1: Novel semantic visual object based applications

Phase	Example Applications
Creation	Video object reuse, virtual scene creation, intelligent capture technology
Storage & Compression	Region of interest coding, spatial/temporal scalable visual object coding, very low bit rate object model based coding
Consumption & Viewing	Semantic transcoding, video summarisation, content based interaction, intelligent searching & browsing
Transmission	Increased error protection for semantically important objects, event triggering based on detection of semantic objects

end life cycle of digital visual data. The same conclusion can also be drawn about audio content, however this research focuses on visual data only.

By processing visual data in terms of the encapsulated semantic objects (see Fig. 1.1), improvements can be made to a plethora of applications within the realm of video/image processing and analysis. These include browsing, searching, video summarisation, transcoding, region of interest compression and scalable compression. This is already evident to users of on-line digital image and video libraries, where the growing use of manual annotation (popularised as "tagging") of the semantics of the content considerably aids searching and browsing [17][18]. However, such manual annotation is costly, tedious and inconsistent. Unfortunately, automatic semantic segmentation (the process of decomposing the scene into unique semantic visual objects) and annotation is an impossible task to solve in a generic fashion. These could be considered the principal reasons why semantic object based processing is not pervasive. Fundamentally, automatic segmentation and annotation is an ill-posed problem. Even from a human perspective, different observers will describe the same scene in different ways. In fact, even the same observer will view the same scene with different levels of granularity based upon their motives for viewing. As will be shown in Chapter 2, the problem can only be made tractable by focusing on specific application scenarios, e.g. sports games, head and shoulder sequences etc. Despite these restrictions, and the likelihood of non-ideal automatic segmentation and annotation, semantic processing still offers the potential to assist and/or make possible novel applications throughout all phases of visual data processing, albeit in constrained domains. Examples of such applications are listed in Table 1.1. The proposed research aims to contribute enabling technologies in this area.

Detecting objects at the point of creation on a mobile device rather than offline on a less computationally constrained computing resource is an attractive proposition as it allows semantics

to be introduced earlier in the digital visual data "life cycle". This has the potential to permit a greater range of novel applications (see Table 1.1). For example, during a mobile video call, the detection of the most important semantic visual objects (e.g. human faces), would allow higher quality encoding or increased error robustness for these objects. This is not possible if the object detection does not take place on the mobile device. Unfortunately, even when the solution space is constrained, semantic object segmentation algorithms are inherently computationally complex. On a mobile device this computational complexity increase diminishes real time performance and increases power consumption. This is highly undesirable as mobile devices suffer from a variety of limitations as will be described in Section 1.2.3. There is a clear conflict between the mobile device hardware limitations and the requirement for robust segmentation algorithms for semantic video processing. This provides strong motivation for innovative research in this domain.

1.2.2 Multimedia Content Delivery to Mobile Devices

The evolution from the analog wireless infrastructure to 2G systems, was principally driven by a need for improved mobile personal communications. Widespread advances in areas including voice quality, hand-off speed, spectrum usage efficiency, error robustness, etc. allowed 2G mobile communications to be very successful in handling voice and low end data. Furthermore, the move to 2G systems also enabled new services such as SMS, fax, Wireless Application Protocol (WAP), etc [19]. These additional services provided extra non-voice based revenue streams for network operators and useful services for end users. However, with bandwidth limited to 9.6 Kilobits per second (Kbps), bandwidth intense services such as video conferencing or video on demand were not feasible. The emerging 3G standard, which uses digital packet switched technology and can achieve data transfer rates up to 2 Megabits per second (Mbps) is an attempt to address this issue. Whilst high data rates are important, it is only one issue when carrying multimedia content on very lossy wireless communication channels. Multimedia content can tolerate some loss, however the time sensitivity of delivery presents unique challenges. Fast hand-off, low latency and minimal packet loss are fundamental to the success of streaming multimedia [20], and as such underpins current research in the area. Active research topics include quality of service, mobility support, signalling protocols, network survivability etc and the impact these have on power consumption on the mobile device [21][22][20][23][24]. In the longer term outlook, research has begun on Fourth Generation (4G) systems, a principal feature of which is the seamless integration of multiple communication technologies including Global System for Mobile (GSM), wireless LAN, bluetooth

etc. Another important feature of 4G systems is the provision of services based upon contextual awareness (location, environmental, health, activity). Clearly, progress in wireless infrastructure is vital for the successful delivery of next generation high resolution multimedia content, however, this is outside the scope of this research and will not be considered further in this thesis.

1.2.3 Multimedia Processing on Resource Constrained Mobile Devices

Multimedia processing is generally considered one of the most computationally demanding tasks, due to elaborate resource intense algorithms coupled with a requirement for high throughput performance, frequently with a real time constraint. Yet a mobile device suffers from a variety of limitations, including low computational capabilities, low memory capacity, short battery life and strict miniaturisation requirements (size, weight, screen size etc). Therefore, the computational demands associated with multimedia processing on a mobile device have the highly undesirable effects of reducing real time performance and increasing power consumption on the device. This issue is further exacerbated by the trend for the mobile device to act as a point of convergence for multiple microprocessor intensive applications. Furthermore, as was demonstrated by the failure of initial WAP services, users are unwilling to tolerate degradation in the overall application experience just for mobile convenience [25][26].

The system designer of a mobile device is now presented with the unenviable challenge of implementing high resource usage algorithms on low throughput and resource limited hardware with short battery life. In particular, as further elaborated in Chapter 2, the short battery life places a major constraint on mobility, since the rate of battery discharge in a mobile device governs the operating time of the device between recharges. In addition, the battery also has a direct impact on the overall size and weight of the device. Unfortunately, the intuitive approach to tackling short battery life by improving battery technology is providing limited results, with battery capacity only improving by 5% each year [7]. Therefore, a more fundamental approach is required, which tackles power consumption as the cause of the short battery life. These factors, coupled with the overarching impact of power consumption on circuit timing performance, silicon resource usage and electronic component reliability have brought energy efficiency to the forefront of design in recent times. Energy efficient design is discussed in detail in Chapter 2.

1.3 Research Motivation and Work Programme

Considering that there is huge consumer demand for novel multimedia applications on mobile devices, the issues outlined in Section 1.2 are unfortunate, particularly as semantic object based processing could be considered an enabling technology toward that goal. This provides the motivation for this research to explore viable implementation options for semantic object processing on mobile devices. The proposed segmentation solution avoids attempting deep automatic interpretation of digital visual data, and focuses instead on mid-level semantic objects. This approach is taken as there is an emerging belief that mapping low level pixels to (frequently well defined) semantic objects is a considerably easier task than a direct mapping from pixels to diverse user semantics [27][28][29].

1.3.1 Novel Face Detection and associated hardware acceleration

In order to demonstrate a robust semantic video object segmentation solution, the problem is constrained and focused on the segmentation of the human face as a fundamental semantic object of benefit to users of mobile devices. As Chapter 2 will demonstrate, there are many approaches to face detection. A fundamental goal of this research is to find solutions which are suitable for deployment on a resource constrained mobile device and this has a major influence of the choice of algorithm. The proposed novel solution (see Chapter 5), employs an Evolutionary Artificial Neural Network (EANN). During regular face detection, the computational complexity associated with the Artificial Neural Network (ANN) phenotype evaluation is offloaded from the host processor to a dedicated hardware accelerator. As will be demonstrated in Chapter 8, this improves energy efficiency and real-time performance. This approach also offers the scope to be retargeted to classify extracted features for other semantic processing tasks. In addition, owing to the widespread deployment of ANNs in a broad spectrum of classification, perception, association and control applications [30], the core could be re-deployed for a number of applications on a mobile device. For example, by modifying the software, the same core that accelerates face detection could be reused as an accelerator for advanced Artificial Intelligence (AI) for computer gaming [31]. This scenario along with a ARM microprocessor based hardware integration framework is depicted in Fig. 1.2. However, applications other than EANN-based face detection are outside the scope of this research.

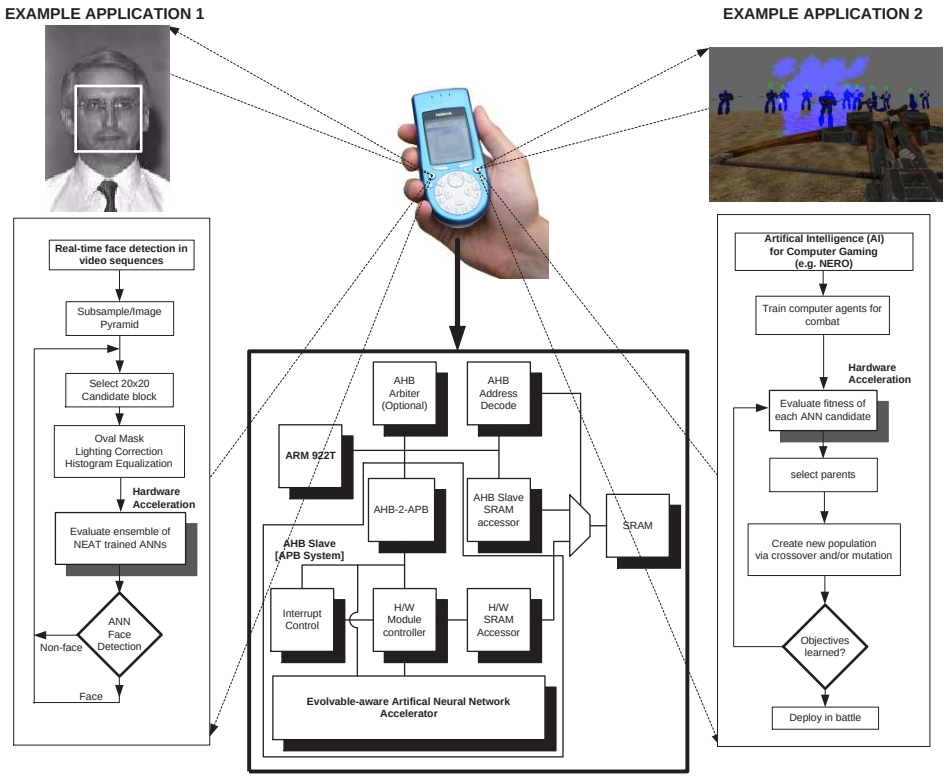


Figure 1.2: Example of multiple applications leveraging a hardware ANN accelerator

1.3.2 Hardware Acceleration of Motion Estimation

Motion Estimation (ME) is a fundamental enabling technology widely used in video compression and video processing tasks. One such video processing task is tracking semantic video objects (such as faces) in video sequences. In the case of face detection, such tracking can be used to improve detection rates and/or reduce computational expense in multi-resolution face detection algorithms. Furthermore, it is generally accepted that ME consumes 40%–80% (depending on the configuration) of the total computational resources required in modern video encoders [32][33]. The re-usability potential, coupled with the high computational expense, makes ME a very suitable candidate for low power hardware acceleration on a semantic video processing enabled mobile device. Chapter 7 describes the proposed ME architecture, which in this research is principally employed to reduce the computational expense of the face detection algorithm described in Chapter 5. With the target deployment of a mobile platform, the architecture is designed with low power operation as the principal design constraint. This goal is achieved by exploiting inherent algorithmic and data redundancies.

1.4 Research Objectives

The research goals of this thesis can be summarised as follows:

1. To design a face detection algorithm suitable for hardware acceleration
2. Evaluate the proposed face detection algorithm against prior art proposed in the literature.
3. To design and implement an energy efficient hardware accelerator for EANN.
4. To design and implement a flexible motion estimation core which can be used in the encoding of binary shape associated with the segmented face .
5. To evaluate both hardware architectures against prior art within a fair benchmarking and normalisation framework.

1.5 Thesis Structure

The thesis is structured as follows:

- **Chapter 1 – Introduction**

  The introduction (this chapter) outlines the context and motivation for the research conducted in this thesis.

- **Chapter 2 – Theoretical Background**

  The theoretical background chapter elaborates in more detail the technical context for the research. The general topics covered are digital video processing (digital video compression & semantic video object segmentation) and low power design.

- **Chapter 3 – Face Detection: A Review of Popular Approaches**

  This chapter firstly presents a thorough review of the prior art in face detection algorithms. Conclusions are drawn from this review and a novel algorithm proposed.

- **Chapter 4 – A Novel Face Detection Training Algorithm**

  The proposed novel face detection training algorithm is described in detail. Comprehensive details are presented for various training runs.

- **Chapter 5 – Software Implementation of Trained Face Detection Algorithm**

  This chapter describes the implementation of the trained EANN face detection algorithm in

software. Software profiling and optimisations are discussed. Face detection performance benchmarking comparisons against prior art are then presented.

- **Chapter 6 – Hardware Acceleration of EANN**

  A review of ANNs and EANN is firstly provided. The proposed hardware architecture for the EANN accelerator is subsequently described in detail. This is followed by evaluation against the prior art.

- **Chapter 7 – Hardware Acceleration of Motion Tracking**

  This chapter initially reviews related research in the field of motion tracking. A detailed description of the proposed motion estimation/tracking hardware architecture is presented. This is followed by evaluation against the prior art.

- **Chapter 8 – Conclusions & Future Work**

  This chapter provides a summary of Chapters 3,4,5,6 & 7, outlining the key contributions. This is followed by a discussion which suggests avenues for future expansion of the research.

1.6 Summary

The growing market for mobile phones is a reflection of how ubiquitous battery powered mobile devices have become in today's society. Multimedia functionality is seen as a vital component in the current and future success of these devices. However, inherently resource intensive multimedia processing algorithms present numerous implementation challenges due to the limited computational resources available. In the future these challenges will be exacerbated by a consumer demand for semantic content aware multimedia processing algorithms. This research aims to help in this regard with contributions to the fields of semantic video object segmentation and dedicated video processing hardware. Specifically a novel face detection algorithm and energy efficient hardware architectures for ANN and ME acceleration are proposed in this research.

CHAPTER 2

Technical Background

This chapter reviews issues in digital video compression, semantic video object segmentation, implementation options and energy efficient design principles in order to provide context for the research carried out. Previously in Chapter 1 it was highlighted that the rapid progress in digital video compression algorithms is a fundamental underpinning technology partly responsible for the ubiquity of high quality digital video on both mobile and tethered computing platforms. An overview of these video compression algorithms/tools along with their associated deployment in industry standards is provided in Section 2.1. Section 2.1.2 discusses how these video compression algorithm and tools can be extended to process content in terms of encapsulated semantic objects, which as was shown in Chapter 1, can facilitate applications allowing a plethora of benefits for the end user. A review of algorithms for the extremely challenging problem of object segmentation is given in Section 2.2. The contributions of this thesis within a semantic video processing context are outlined in Section 2.3. A selection of implementation options for advanced video processing on computationally constrained mobile devices is presented in Section 2.4. A vital consideration for implementation of advanced algorithms on a mobile device is energy efficiency. As such, a detailed review of energy efficient design principles is given in Section 2.5. This includes an overview of the sources of power consumption, as well as common techniques used to reduce power consumption.

2.1 Digital Video Compression

Before considering digital video[1] compression techniques or semantic video object processing algorithms, the fundamental properties and characteristics of raw unencoded digital video require explanation. A raw unencoded digital still image can be considered to be a "snap shot" of a natural real world scene, which results in a two dimensional rectangular or square matrix of discrete digital samples capturing the colours present. It is assumed that the real world scene contains smooth continuous colour tones with bandwidth limited edges (these are characteristics which can be exploited in compression algorithms). Each resultant digital sample is a representation of the colour in the scene at a particular location. The colour can be represented in a variety of different schemes (i.e. different colour spaces). In the most direct scheme each sample consists of Red, Green and Blue (RGB) components. Typically, 24 bits per RGB triplet is considered to give adequate dynamic range, although it is not uncommon for a greater number of bits to be used. For example, in some High Definition Television (HDTV) encoder/decoder datapaths and medical imaging storage applications ten or twelve bits are used. As with all analogue to digital sampling, Nyquist sampling theory must be considered, although the human psychovisual system is quite tolerant to many aliasing effects[2] [34]. The size of the rectangular sampling matrix or frame is application and transmission bandwidth dependent. For example, on mobile devices with limited screen size and limited transmission bandwidth, standardised resolutions such as 352×288 (referred to as Common Intermediate Format (CIF)) or 176×144 (referred to as Quarter Common Intermediate Format (QCIF)) samples or pixels are popular, whilst at the other end of the consumer market, European PAL based Standard Definition Television (SDTV) and HDTV have typical active resolutions of 720×576 and 1920×1080 respectively.

Digital video temporally extends digital still image sampling. There is continuous sampling of a scene at discrete time intervals in conjunction with spatial sampling in two dimensions (see Fig. 2.1). The temporal sampling resolution is typically quoted as the number of frames per second. It should be noted that the visual quality of the captured video is heavily influenced by the spatial and temporal resolution. A high horizontal and vertical sampling allows a natural scene to be captured with more fine detail and thus creates a greater sense of realism. A high

[1] In this thesis, unless explicitly stated otherwise, digital video is frequently used to refer to both digital video and digital still images. This generalisation is made as digital video could be considered to be a superset of digital still images. That is, digital video is an extension of digital still images in the temporal domain.

[2] This is not always the case. For example, a familiar aliasing phenomena (present particularly in older films) that is noticeable and quite objectionable, occurs when the spokes of a fast moving waggon wheel appear to move backward. This is caused by temporal aliasing [34].

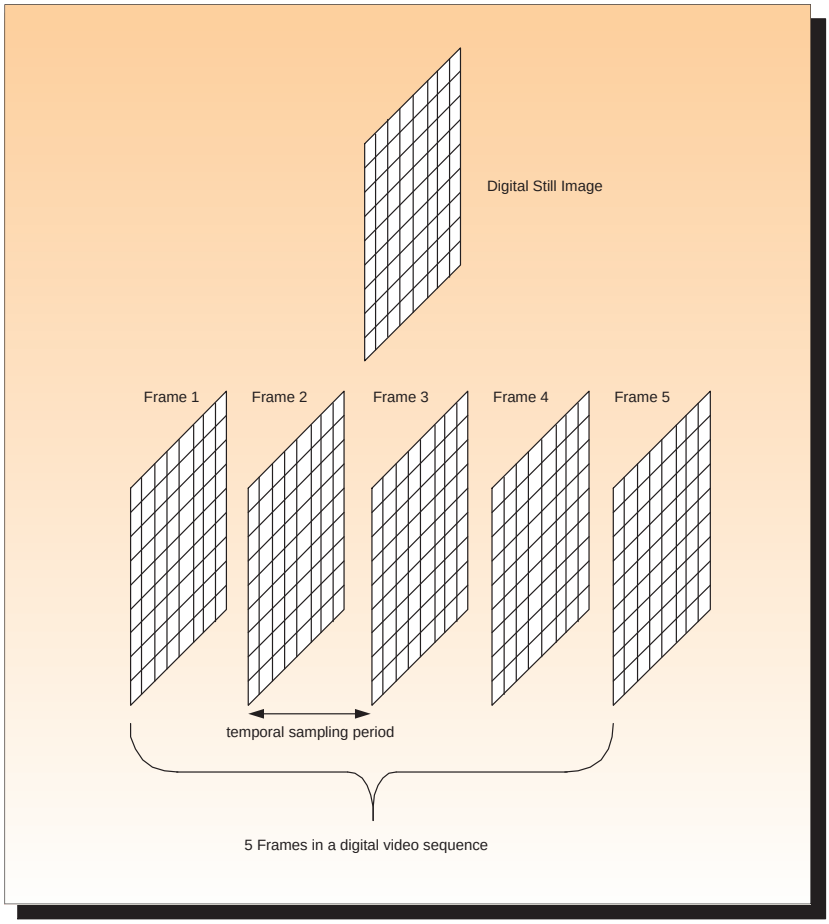


Figure 2.1: An example of a digital still image and video sequence

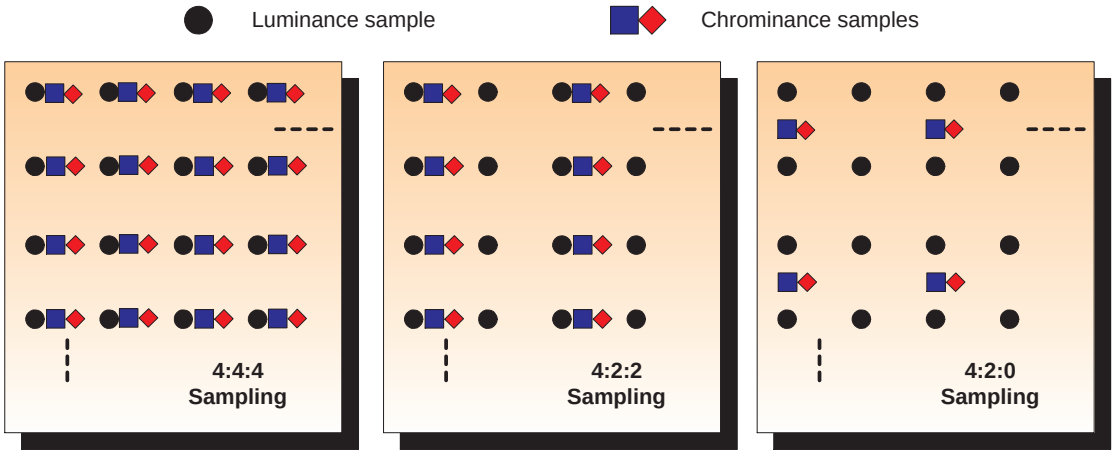


Figure 2.2: Examples of YC_bC_r sampling modes

frame rate (i.e. temporal sampling rate) allows objects within a scene that have high levels of motion to be perceived as having smooth continuous trajectories, whereas if the sampling rate were lower the object could appear as having a disconcerting jerky movement. Increasing the frame rate obviously has a proportional increase in the unencoded bit-rate of the video and this is not desirable in many applications. For example, low bit-rate communications (e.g. webcams, mobile video calls) sometimes use a frame rate as low 5-15 frames per second. However, this can result in jerky motion. The use of 25 to 30 frames per second is generally sufficient to capture moving sequences for viewing on a mobile device, whilst a higher frame of 50 to 60 frames per second is generally required for larger display sizes (e.g. SDTV) to allow fast moving objects have smooth motion. When the trade-off between sampling resolution (both spatial and temporal) and data transmission rates was encountered during the design of early TV transmission standards, a solution termed interlacing was proposed[3]. In this scheme, only half of the frame data (even or odd lines) is transmitted every frame. The reduced data frame is termed an even or odd field depending on the line type that was sent. The scheme doubles the perceived frame rate without doubling the bandwidth. Although, this is not without disadvantages, as distortions are introduced. For the remainder of this thesis it can be assumed that non-interlaced video (also known as progressive video) is being used.

As previously mentioned, the RGB colour space is typically used during the sampling of a natural scene into the digital domain. In addition, it is also used for outputting digital images on Cathode Ray Tube (CRT) or Liquid Crystal Display (LCD) units. In the RGB colour space each component has an equal weighting. However, the human psychovisual system is more sensitive to brightness (luminance) than colour (chrominance). As a result a common processing step is to convert from the RGB colour space into the YC_bC_r colour space. The Y component represents luminance, and C_bC_r are the chrominance components. The benefit of this colour space conversion, is that the chrominance components can be subsampled, thus saving storage and/or transmission bandwidth. Video compression algorithms usually support 4:4:4, 4:2:2 and 4:2:0 luminance/chrominance sampling (see Fig. 2.2 for more details). The 4:2:0 sampling scheme is the most popular for video compression algorithms and results in little or no discernible loss in visual quality and has the benefit of reducing the data by a factor of two.

Regardless of the format (resolution, sampling scheme etc.) chosen, raw unencoded digital video requires enormous storage capacity and/or transmission bandwidth. For example, using

[3] Another motivating factor for interlacing was due to the limited refresh speed of cathode ray tube based TVs of the era

the CIF format with 4:2:0 colour sampling and 25 frames per seconds requires: $352 \times 288 \times 1.5$ bytes $\times 25 = 3.6255$ MBps. Putting this into context, this low resolution format, which is only suitable for a mobile device, requires in excess of 13 gigabytes of storage for one hour of video. This bandwidth requirement is at odds with the characteristics of the target application i.e. extremely limited storage space available on mobile devices, limited bandwidth mobile communications and the relatively high cost of mobile communication data transmission services. As a further example, SDTV has a bandwidth requirement of 216Mbps, so without further processing, a DVD could hold little more than a few minutes of SDTV video. This clearly illustrates the need for efficient compression of video data. Fortunately, this can be achieved by exploiting the many inherent redundancies in the video data representation. The tools used in a video compression system are explained in the following subsections.

2.1.1 A Generic Video Compression System

There are a number of broad categories of inherent redundancies within a raw unencoded video data representation which can be exploited to achieve compression. These inherent data redundancies can be summarised as the following:

- **Spatial Redundancies:**

  There is considerable redundancy between adjacent pixels, such as in large homogeneous coloured regions like a blue sky. In addition, colour transitions are typically gentle, reflecting the way they occur in the natural world.

- **Temporal Redundancies:**

  There is frequently little difference between consecutive frames of video unless considerable motion and/or a scene change has occurred. This can be exploited by only coding and transmitting those regions within the frame which have changed.

- **Statistical Data Redundancies:**

  Some symbols used within the video compression system have a higher probability of occurring than others. Rather than employ an equal word length for all the symbols, if a shorter word length is used for more frequently occurring symbols, an overall reduction in data can be achieved. Morse code is a classical non-video example of where statistical redundancies in a data distribution is used to reduce the bandwidth.

- **Properties of human visual system:**

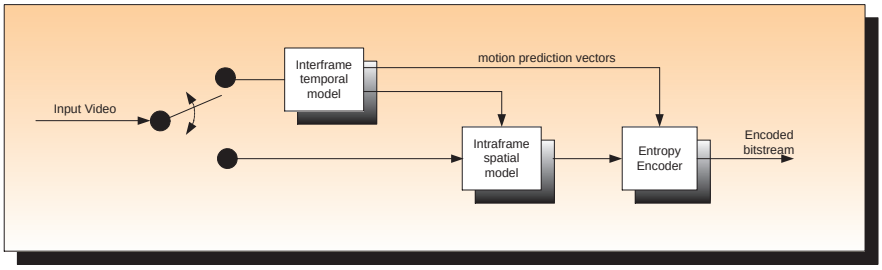
  As previously mentioned, the human visual system is more sensitive to luminance than colour. This is typically exploited by subsampling the chrominance component. In addition, the human visual system is less sensitive to areas of high frequency e.g. sharp colour transitions.

Ideally the compression algorithm applied to the inherent video data redundancies at the encoder would be perfectly reversible at the decoder. This type of compression is known as *Lossless*. Lossless compression is vital for example in computer file compression algorithms, where any loss/corruption would destroy the overall message. Whilst lossless video compression algorithms[4] exist, the compression achievable is small, typically in the order of 2-4x [35]. With the exception of a small number of applications in niche areas[5], this rate of compression is insufficient due to the sheer quantity of video data. Fortunately, from a compression perspective, a perfect reconstruction at the decoder is not always necessary. In many applications, the eye can tolerate a loss. This form of compression is termed *Lossy*, since information is lost during the compression process. The challenge then becomes one of balancing compression efficiency (i.e. size of the resultant bitstream) versus acceptable visual quality for a given application. In the case of mobile applications, it may also be tolerable to accept reduced visual quality for reduced computational complexity in order to improve the real-time performance and/or power efficiency reasons.

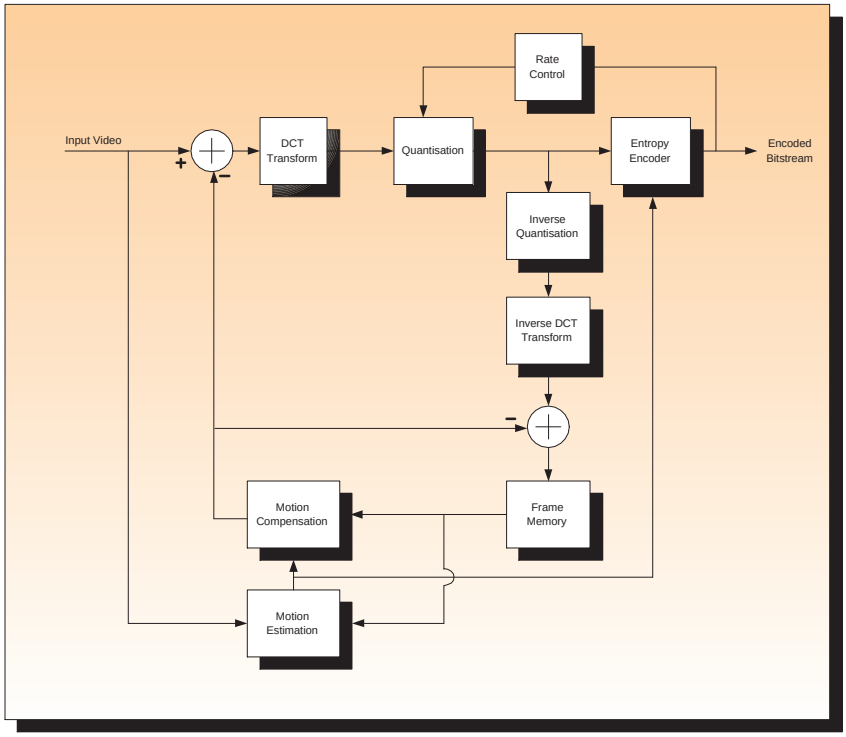
Mainstream video compression algorithms use a combination of interframe and intraframe coding for both lossless and lossy compression (see Fig. 2.3(a)). Intraframe coding exploits spatial and perceptual redundancies and is coded without reference to other frames. Interframe coding exploits temporal redundancies by using previous (and sometimes future) frames. In both cases, further processing, known as entropy coding, is used to exploit the statistical data redundancies. This general video compression framework is typically known as a *hybrid video codec* and is the basis of all modern video compression systems. Fig. 2.3(b) and Fig. 2.3(c) show the principal constituent elements of a generic hybrid encoder and decoder in greater detail. These elements are explained in the subsequent subsections.

[4] Zero mathematical loss of visual quality at the decoder

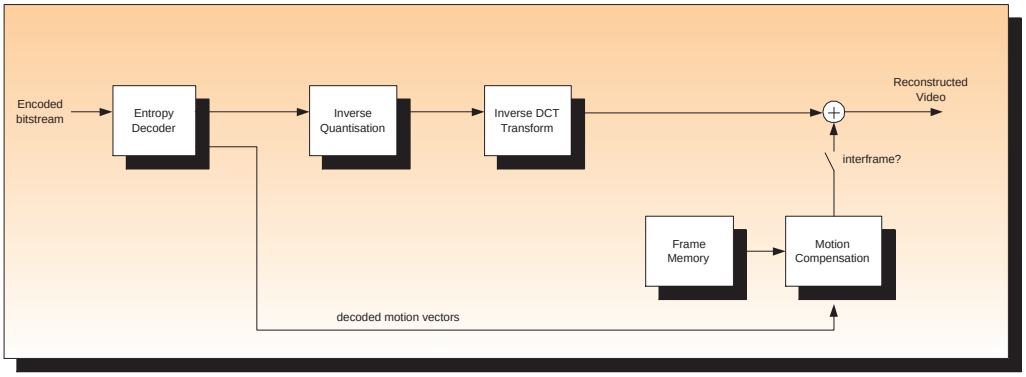
[5] Such niche area applications include compression of video data bus transfers and intermediate storage of video in production studios where quality is imperative



(a) A Simplified Model of a Video Encoder



(b) A Generic Hybrid Video Encoder



(c) A Generic Hybrid Video Decoder

Figure 2.3: A Generic Video Encoder and Decoder

2.1.1.1 Motion Estimation & Compensation

There is typically only a very slight change between successive frames in video. For example, Fig. 2.4(a) and Fig. 2.4(b) show temporally adjacent frames from a test sequences. A temporal prediction model at the encoder & decoder can exploit this redundancy and will provide compression provided the model parameters and any correction terms are less than the raw pixel information. As can be seen in Fig. 2.4, if frame one is subtracted from frame two, the residual energy (shown scaled in Fig.2.4(d) for ease of viewing) contains considerably less detail. It is reasonably intuitive that the residual frame will require less data bandwidth than the original frame 2. In fact, the entropy[6] of the original frame two is 7.15 bits/pixels, whereas that of the temporal prediction residual is 4.38 bits/pixels. In the previous example the temporal prediction model was the most basic available, that is simple frame differencing. More complex temporal models can reduce the entropy further by more accurately capturing the interframe temporal behaviour, which can be attributed to a combination of camera motion & object motion. However, for practical implementation purposes the choice of temporal prediction model is also a trade off between complexity, memory requirements and prediction performance.

A generic temporal prediction model can be considered to consist of *Motion Estimation* and *Motion Compensation*. Motion estimation attempts to establish the motion that has occurred between frames, whilst motion compensation uses the motion vectors generated from the motion estimation process to retrieve the predicted block. There are two main approaches used commonly for motion estimation, gradient descent based algorithms and block-matching algorithms (see Fig. 2.5). Of these, the block matching approach gives acceptable prediction performance whilst using less computational resources when implemented in either software or hardware. The block based algorithm examines each $M \times N$ block in the current frame and finds the associated best matching block within a predetermined or adaptive $\pm S$ pel search range in a reference frame(s). Motion compensation uses the motion vector associated with the best match to retrieve the appropriate block, which is then subtracted from the current block to generate a prediction residual. The prediction residual is then used in subsequent processing. A subtle point that should be noted is that since differences will likely exist (i.e. due to the lossy process) between the "real" previous frame and the decoded previous frame in the decoder, the motion estimation/compensation process does not use the real previous frame but rather uses the decoded frame in the encoder (see Fig. 2.3(b)).

It is also worth emphasising the fact that block based algorithms estimate the motion of a group

[6]Entropy is a measure of the amount of information content in an information source

(a) Frame 1

(b) Frame 2

(c) Frame Difference

(d) Scaled Frame Difference

Figure 2.4: Temporal Redundancy in Video Sequences

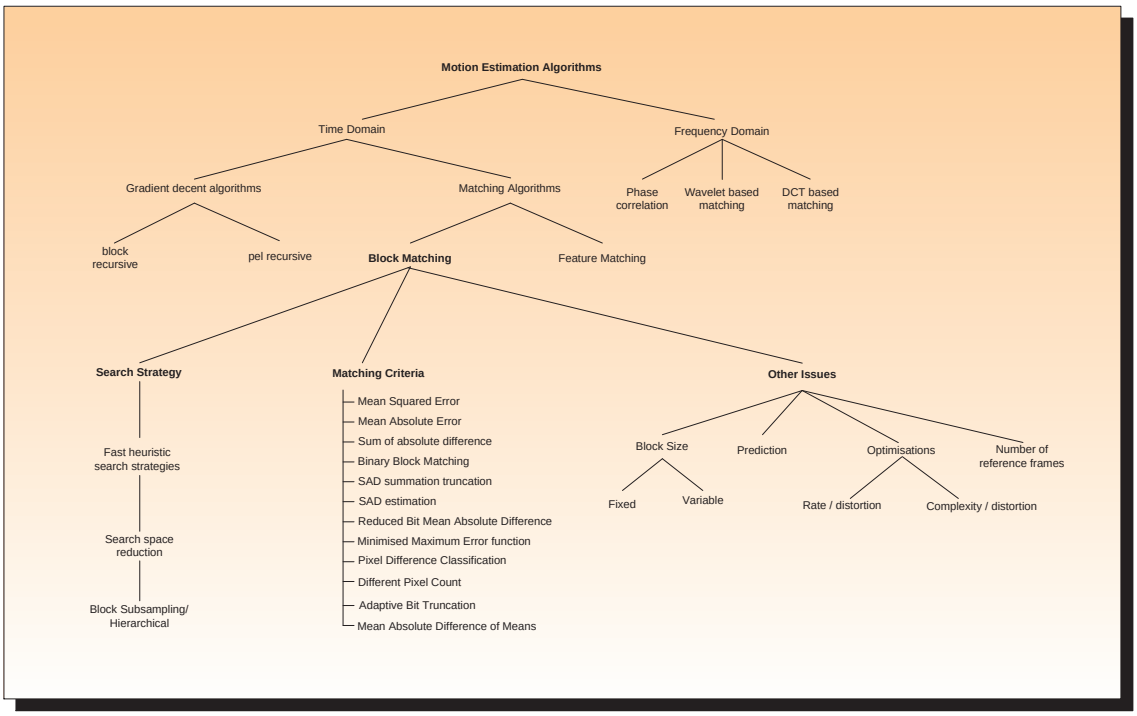


Figure 2.5: Motion estimation taxonomy

of pixels. This generally works quite well, as semantic objects are generally much larger than the typical range of block sizes, thus frequently the block based motion estimation algorithm generates a motion field that is relatively consistent with the trajectory of an object. The block size leads to certain characteristics of the resultant motion vectors and prediction residual. A small block size can give very good prediction results, but is also likely to get caught in local minima which are not consistent with the true motion of the object. Although in video compression, establishing the true motion is a secondary concern compared to energy minimisation of the prediction residual. In contrast, in other video processing applications[7] the true motion is of principal importance. A further issue with a smaller block size is that there is also an increased computational complexity cost in the motion estimation algorithm. However, it could be argued that the greatest issue with smaller block sizes for video compression is that they require a greater number of motion vectors to be added to the bitstream. So although the prediction residual may have less energy, this is counter balanced by the increased bits required to store the motion vectors. The alternative option of using a large block matching size is also not without disadvantages. For example, on occasions it lacks the resolution necessary for complex motion fields in highly detailed blocks (e.g. a block containing the edges of two or more objects moving in differing directions). This issue of block

[7]For example, in frame rate conversion for HDTV decoders (i.e. 50/60 Frames Per Second (FPS) to 100/120 FPS) inaccurate motion vectors lead to poor interpolation results with highly objectionable artifacts.

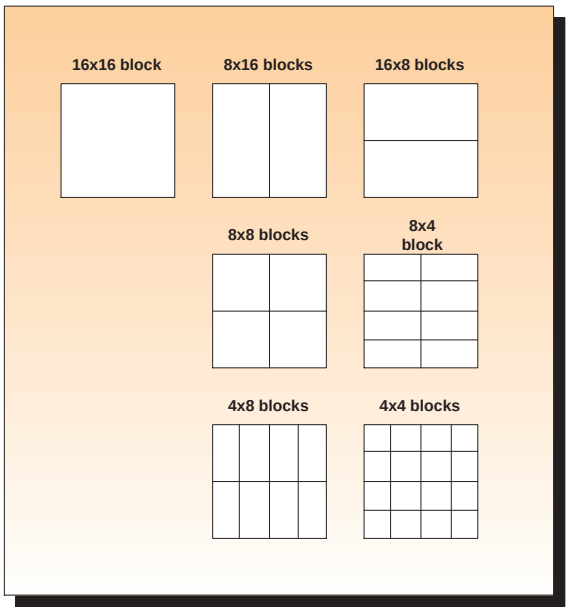


Figure 2.6: Block sizes used in H.264 motion estimation)

size is addressed in newer video compressions standards (MPEG-4 and to a greater extent MPEG-4 part 10 / H.264) through the use of variable block size. H.264 uses a Lagrange optimisation to establish the optimum trade off between a range of block sizes (see Fig. 2.6), the quality of the prediction residual and the cost of the motion vectors.

There are principally two constituent elements in a block matching algorithm; the block matching routine and the search strategy. The search strategy finds appropriate candidate blocks within a search window, whilst the block matching evaluates a distortion metric (level of similarity) between each candidate block in the search window and the current block in the current frame. The search strategy and block matching typically operate on just the luminance component, with resultant motion vectors scaled for the chrominance blocks according to the chrominance subsampling strategy employed. A wide variety of matching criteria can be used, these include Mean Squared Error (Eqn. 2.1), Mean Absolute Differences (Eqn. 2.2), Sum of Absolute Differences (SAD) (Eqn. 2.3), Binary Block Match (Eqn. 2.4), SAD summation truncation, SAD estimation, Reduced Bit Mean Absolute Difference, Minimised Maximum Error function [32]. The matching criteria is a complexity/prediction performance trade off and represents a vital computational complexity decision as the distortion metric is typically inside deep inner loops of search strategies. For example, in the case of binary valued pixels (single bit representation as opposed to 8 bits) the SAD calculation is simplified. This allows the subtraction operation in Equation 2.3 to be replaced by a single bit XOR, since the difference between two pixels will be either 0 or 1. The absolute

function is also implicit in the XOR function, because the XOR cannot give a negative result. The operation reduction coupled with less data (1 bit instead of 8 bits per pixel, generated through a binarisation/quantisation process) is beneficial from the point of view of computational complexity. However, the lack of pixel value granularity typically leads to degradation in the quality of the motion vectors, which causes the prediction residual to have more energy. Many distortion metrics were evaluated by Kuhn in terms of image quality (PSNR) and implementation results (area, throughput, power) [32]. He found that the SAD metric gave the optimum trade off between complexity and efficiency/quality.

$$MSE\left(B_{curr}, B_{ref}\right) = \frac{1}{M \times N} \sum_{i=1}^{i=M} \sum_{j=1}^{j=N} \left(B_{curr}\left(i,j\right) - B_{ref}\left(i,j\right)\right)^2 \tag{2.1}$$

$$MAD\left(B_{curr}, B_{ref}\right) = \frac{1}{M \times N} \sum_{i=1}^{i=M} \sum_{j=1}^{j=N} \left|B_{curr}\left(i,j\right) - B_{ref}\left(i,j\right)\right| \tag{2.2}$$

$$SAD\left(B_{curr}, B_{ref}\right) = \sum_{i=1}^{i=M} \sum_{j=1}^{j=N} \left|B_{curr}\left(i,j\right) - B_{ref}\left(i,j\right)\right| \tag{2.3}$$

$$\text{Binary SAD}\left(B_{curr}, B_{ref}\right) = \sum_{i=1}^{i=M} \sum_{j=1}^{j=N} \left(B_{curr}\left(i,j\right) \otimes B_{ref}\left(i,j\right)\right) \tag{2.4}$$

In Eqn. 2.1 to Eqn. 2.4, B_{curr} is the block under consideration in the current frame and B_{ref} is the block at the current search location in the search frame. The block size is $M \times N$.

Once the distortion metric is calculated between the current block and reference block, the process repeats until all of the search positions defined by the search strategy within the search window are examined. The block match which gave the minimum distortion metric is deemed to be the most suitable match and used for subsequent processing. There are a wide variety of search strategies that can be used. The choice is typically a complexity/performance trade off, with the following being the broad options available [32]:

- Exhaustive full search algorithm

  This searches every position within the search window and as such always gives the best results, but it is also very computationally expensive. However, the regularity of the search positions is suitable for hardware.

- Fast exhaustive search

  This search strategy eliminates non-optimal search positions while maintaining optimal mo-

tion vectors through early termination of distortion metric calculations. Sometimes this process is called a SAD step cancellation search strategy.

- Fast heuristic / logarithmic search strategies

  This approach uses the assumption that the distortion metric monotonically increases moving away from the minimum point to reduce the number of search positions. However, there is the distinct possibility of this search strategy getting stuck in a local minimum, which in turn yields a higher energy prediction residual. The three step search is an example of this approach.

- Hierarchical or multi-resolution search strategies

  An initial search takes place in a low resolution version of image, progressively higher resolution searches take place in the parts which exhibit the best promise. Again this approach reduces the number of search positions relative to the full search and compared to the logarithmic search strategies, the hierarchical nature can help to avoid local minima due to a low pass filtering effect. However, local minima issues are still possible for small regions which disappear during the sub-sampling process.

- Zone based search strategies

  These search strategies employ numerous stopping thresholds, which can be advantageous in a rate/distortion sense. For example, a spiral search coupled with multiple thresholds defined for different search radii could terminate calculations early if one of the predetermined thresholds was reached.

- Motion vector prediction and dynamic search window size

  These two techniques can be applied to most search strategies. A prediction of the motion vector can be used to seed the search strategy. In the ideal case, if the prediction is accurate enough (i.e. below a specified threshold), no further processing is required. Otherwise a conventional search is carried out around the motion vector predictor. If for example, a logarithmic or zone based search strategy is then used, the motion vector prediction is still likely to reduce the overall number of operations necessary to give a satisfactory result and/or improve the quality of the prediction residual. In a similar fashion, a dynamic search window can be used to reduce the number of operations by constraining the search window in areas of predicted low motion. Typically spatial and/or temporal correlation methods are used to estimate the motion vector predictor and search window size.

2.1.1.2 Transform Coding

Most mainstream video compression systems use transform-based algorithms, which literally transform the spatial information (i.e. the pixels or a motion compensated prediction residual) into an alternative representation which is more amenable to compression. This alternative representation is almost always in the frequency domain. Whilst both the spatial and transform domain are equivalent, spatial redundancies and the sensitivities of the human visual system can be more easily exploited in the frequency domain. This is because energy in a natural scene is concentrated in low frequencies and in addition the eye also has greater sensitivity to lower frequency content. Therefore, once the spatial data is transformed into the frequency domain, subsequent processing can direct greater coding resources to those frequencies which are the more perceptually important.

Many transforms have been suggested for image and video compression including Karhunen-Loeve Transform (KLT), Singular Value Decomposition (SVD), Discrete Wavelet Transform (DWT), Fourier transform, Discrete Cosine Transform (DCT) etc. The ideal transform compacts the largest amount of energy into the smallest number of coefficients. The KLT is optimum in terms of packing the greatest amount of energy into the fewest transform coefficients. A contributing factor for this performance is that the KLT calculates the optimal transform matrix on a block by block basis. However, for practical implementation purposes the choice of transform is a trade off between energy compaction, reversibility and computational complexity. The KLT is excessively computationally demanding to be practical and in addition requires considerable overhead to be transmitted to the decoder. The DCT is a close approximation to KLT in terms of performance and is reversible[8], but requires considerably less computational resources and no overhead communication to the decoder. For these reasons the DCT is frequently used in image and video compression systems.

$$F(k,l) = \frac{2}{\sqrt{N^2}} C(k)C(l) \sum_{x=0}^{N-1} \sum_{y=0}^{N-1} f(x,y) cos \left[\frac{(2j+1)\, k\pi}{2N} \right] cos \left[\frac{(2i+1)\, l\pi}{2N} \right] \tag{2.5}$$

[8] In theory the DCT is perfectly reversible in the IDCT, but due to issues such as finite precision arithmetic, mismatches can occur. MPEG-2 and MPEG-4 tackle this issue using techniques such as oddification and LSB toggling [35]. H.264 avoids the problem by using an integer based transform, which is exactly reversible

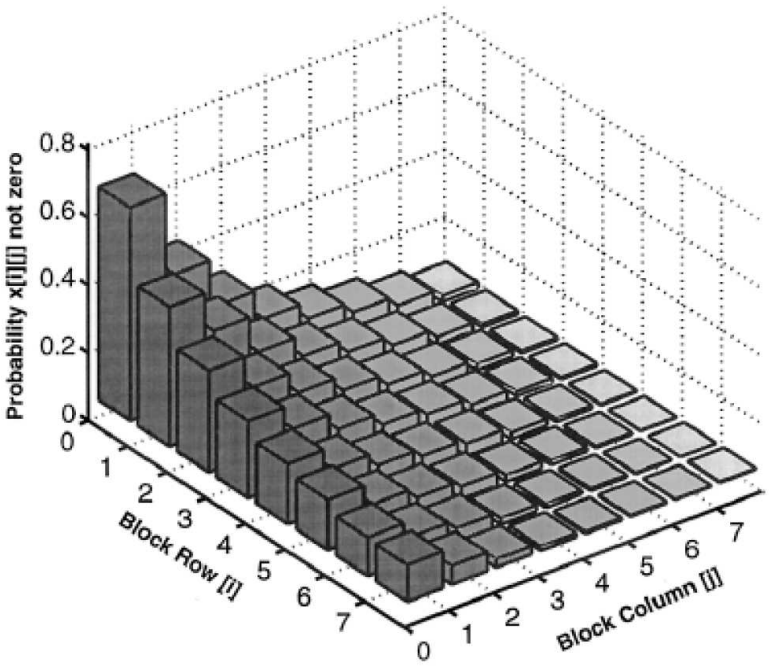


Figure 2.7: Probability of DCT coefficients being nonzero (Source [36])

$$f(k,l) = \frac{2}{\sqrt{N^2}} \sum_{l=0}^{N-1} \sum_{k=0}^{N-1} C(k)C(l)F(k,l)cos\left[\frac{(2j+1)\,k\pi}{2N}\right] cos\left[\frac{(2i+1)\,l\pi}{2N}\right] \tag{2.6}$$

The generalised form for the 2D DCT transform and the Inverse Discrete Cosine Transform (IDCT) is given in Eqn. 2.5 and Eqn. 2.6 respectively. In both cases k, l = 0, 1, . . . ,N - 1 and F(k, l) denotes the DCT coefficient at coordinate (k, l). Whilst f(x, y) is the input pixel at (x, y) with C(0) = $\frac{1}{\sqrt{2}}$, and C(k) = C(l) = 1 otherwise [37]. The DCT can be calculated for any rectangular array of pixels, though in video compression an 8×8 matrix is generally used. This results in 64 coefficients, each of which can be considered to give a measure of how much content is contained in that frequency. For example, the top left corner of the coefficient matrix represents the average (or DC) value of the block, whilst the bottom right hand corner contains content having the highest horizontal and vertical frequency present in the block. As natural scenes tend to change slowly, it is intuitive that the energy is concentrated in the DC value and lower frequency coefficients with the higher frequency coefficients typically having much smaller values. The characteristic is demonstrated in terms of the probability of non-zero coefficient values in Figure 2.7 [36]. For a more thorough mathematical treatment of the DCT the reader is referred to the extensive research in the literature [37]. Extensions and variants of the DCT are also common, such as the integer based DCT found in the MPEG-4 part 10 / H.264.

2.1.1.3 Quantisation

As discussed previously, the DCT or indeed other transforms do not compress the source image/frame data, rather the compression is achieved through the use of quantisation and entropy coding. The purpose of the quantiser is to remove transform coefficients which have less perceptual contribution to the quality of the reconstructed block. This is an inherently lossy process, data discarded through coarse quantisation at this phase cannot be recovered later at the decoder and as such the compression achieved is a trade off with the quality of the reconstruction at the decoder. There are principally two main types of quantisers, scalar quantisers which operates on a single coefficient and vector quantisers which processes multiple coefficients. Vector quantisers use codebooks and can be used to quantise transform coefficients or raw pixel values. For example, in the GIF standard a vector quantiser is used to reduce the number of colours in a block. However, by far the most frequently used quantiser in video codecs is the scalar quantiser. Whilst nonlinear scalar quantisers exist (e.g. Lloyd-Max quantiser, entropy constrained quantiser, etc), the uniform linear scalar quantiser (which is also the simplest) is frequently found to be the most effective. A uniform linear scalar quantiser has equal step sizes and the reconstruction value is set to the centroid of the step. A larger step size reduces the precision of the resultant quantised coefficient and thus also reduces the number of bits necessary to store the coefficient. In cases where the coefficient has a low initial value, quantisation can make the coefficient's value null. Normally the quantised coefficient is scaled by the step size and rounded before further processing in order to reduce the entropy. A simple extension of the uniform scalar quantiser is to have a "deadzone" (i.e. an input region which has output quantised values of zero) around zero. This is useful as it can force a greater number of small valued high frequency coefficients to zero. A quantiser with a deadzone typically has a variable step size and generally the step size is fixed for a quantiser without a deadzone. A fixed step size is generally used for DC coefficients whereas a variable step size is useful for AC (i.e. non DC) coefficients. Different codecs employ different configurations of stepsize and deadzones and this is covered extensively in the literature [38][35][39].

2.1.1.4 Entropy Coding

In the case of an inter-frame, motion estimation/compensation is used to make a temporal prediction generating motion vectors and a prediction residual. The motion compensated residual or in the case of a intraframe, the raw pixels, undergoes DCT transformation which results in DCT coefficients that are subsequently quantised. The motion vectors (for interframes) and quantised

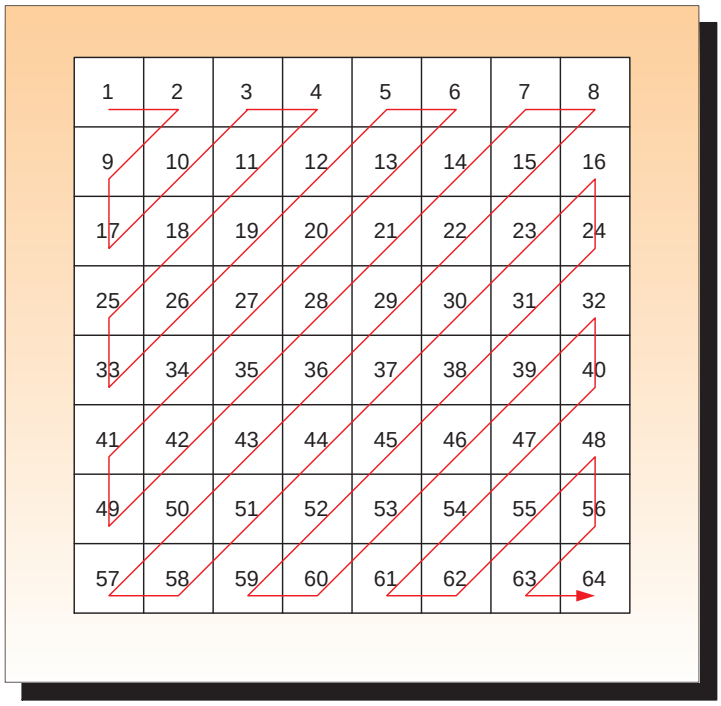


Figure 2.8: Zig-zag scanning of quantised DCT coefficients

coefficients along with other encoder configuration settings (e.g. resynchronisation points, macroblock headers, etc.) must be added to the final bitstream in order for the decoder to be able to reconstruct the frame. It is feasible to think that each of these information units or symbols (as referred to in information theory) can be added directly in byte format to the bitstream without further processing. However, by exploiting the statistical probabilities of the occurrence of these symbols greater compression can be achieved. This is essentially the purpose of *Entropy Coding*, to assign shorter code words to the more frequently occurring symbols. It is a lossless process, so no further degradation is introduced. The theoretical minimum number of bits needed to code an information source (termed the entropy) is given in Equation 2.7. This equation demonstrates that the higher the probability (shown as P in Eqn. 2.7) an event has of occurring, the lower the number of bits required to store the symbol.

$$H(x) = \sum P_i Log_2 \frac{1}{P_i} \tag{2.7}$$

Prior to entropy coding, the quantised DCT coefficients are reorganised into a format that lends itself more readily to compression. After quantisation many of the DCT coefficients are zero. Instead of each zero coefficient undergoing entropy coding, a simple technique known as *Run Length Coding* is used that replaces the number of consecutive values (e.g. zero) with the

value and the number of times it repeats. For greatest compression from run length coding it is imperative to maximise the number of consecutive zeros. Whilst the quantised coefficients can be read out in a raster scan order, the runs of zero valued high frequency coefficients would be frequently interrupted by non-zero low frequency coefficient values. As a result a *Zig-zag* scan is used to maximise the length of consecutive zeros. This scanning order is shown in Fig. 2.8.

Following zigzag scanning and run length coding, the DCT coefficients along with the motion vectors and other configuration parameters undergo entropy coding. The two most widely used entropy coding algorithms in image/video compression are *Huffman Coding* and *Arithmetic Coding*. In its simplest form Huffman coding can be considered to consist of two phases: source reduction and codeword reconstruction. Source reduction is an iterative process of combining symbols with increasing probability, until only two combined symbols remain. Starting from the final two symbols and working backward to the original symbols, codeword reconstruction allocates codes of increasing length at each symbol combination node in such a manner that each symbol is uniquely decodeable [34]. In reality, in video codecs, the Huffman codes are precalculated for generic video material (e.g. differentially coded motion vectors, transform coefficients, etc.) and stored in look up tables [35]. Huffman coding is optimum if the symbols have probabilities that are negative exponents of two, if not the theoretical minimum number of bits cannot be reached. One way of improving the performance in such a case is through the use of arithmetic coding. This codes groups of symbols together, so overall a fractional number of bits can be assigned per symbol. Essentially arithmetic coding works by continuously updating probability codes, and outputting bits when the most significant bits will not change further [38]. Performance improvements of 5%-15% have been observed when using arithmetic coding compared to Huffman coding, although the computational complexity does increase [38].

2.1.2 Semantic Video Object based Compression

Traditional video processing algorithms such as MPEG-1, MPEG-2, etc treat visual data as a sequence of rectangular frames [38][34]. Whilst advances in exploiting spatial and temporal correlation in image and video data have lead to ever increasing compression rates and improved reconstructed quality, the approach is inherently unaware of the semantic structure of the visual data [40][41]. In contrast, once a segmentation mask is provided for each semantic object, ISO/IEC MPEG-4 Core profile provides a complete semantic multimedia compression framework [42]. This allows a video sequence to be compressed in terms of one or more semantic video objects,

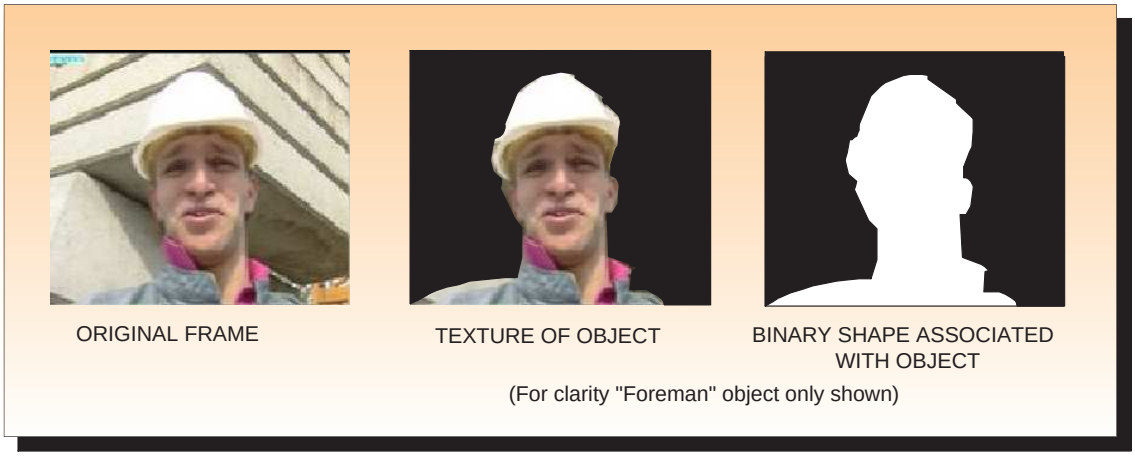


Figure 2.9: MPEG-4 Video Objects

each with associated texture and alpha (i.e. shape) information. The object based compression paradigm is much more powerful than previous generations of MPEG standard because as was noted in Chapter 1, it potentially allows improvements in a wide variety of applications within the realm of video/image processing and analysis, including browsing, searching, video summarisation, transcoding, region of interest compression, error protection and scalable compression.

Shown in Fig. 2.9 is a snapshot of the "Foreman" test sequence with the associated segmented texture and alpha information. Similar to the way that a snapshot of video is termed a frame, a snap shot of a video object is called a Video Object Plane (VOP). The associated shape information can be either binary or greyscale (e.g. shown in the binary form in Fig. 2.9). A binary shape mask indicates whether each pixel is part of an object, whereas greyscale shape information allows an object to have varying levels of transparency, which is useful for blending an object (particularly the edges of the object) onto a background. A snapshot of the binary alpha information is termed a Binary Alpha Plane (BAP), whilst a coding unit or block with the BAP is frequently referred to as a Binary Alpha Block (BAB) and is normally sized 16×16 in MPEG-4. There are three distinct types of BAB: transparent, opaque and boundary. An opaque BAB is fully inside the object and appears as white in Fig. 2.9. A transparent BAB is a BAB that is completely outside the object. A boundary BAB is as suggested by the name, a BAB on the edge of the object and incorporates pixels which are fully inside and fully outside the object.

To facilitate the added object based functionality, modifications to the conventional hybrid codec (see Fig. 2.3(b)) are necessary. In particular, variants of the traditional frame based DCT and motion estimation tools are used [43]. The video object shape information is compressed in a dedicated shape encoder, which principally consists of binary motion estimation, a sub-sampling

unit and context arithmetic encoding [44]. The shape coder will be explained in more depth in the following section. The DCT transformation is replaced by the Shape Adaptive Discrete Cosine Transform (SA-DCT). Using the alpha information, the SA-DCT transforms only pixels which are part of the object. Transparent BABs are ignored, while the SA-DCT reverts to a regular 8×8 DCT transform for fully opaque BABs. The shape adaptiveness of the transform is used on boundary BABs, where pixel shifting is used to facilitate a variable point DCT. Motion estimation must also be modified for object based processing. The most direct approach simply uses pixel padding for boundary or opaque BAB texture blocks. The alternative approach of *polygon matching*, only evaluates the distortion metric on those pixels which are part of the object. Although mentioned in Chapter 1, it is worth reiterating that the segmentation of the objects is not defined by MPEG-4. Nevertheless, the availability of MPEG-4 highlights that if semantic video objects are available, efficient compression and transportation of those objects is already possible. MPEG-4 object based processing does however introduce an additional computational cost over the non-object based MPEG-4 simple profile.

2.1.2.1 MPEG-4 Shape coding

The alpha information represents additional overhead which must be communicated to the decoder, therefore it is logical that this information should also be compressed. A binary shape encoder is used regardless of whether the alpha information is binary or greyscale. This is because the shape of the greyscale alpha map is encoded using the binary alpha encoding flow, whilst the greyscale transparency information is coded in a transform based process. Only binary shape encoding is discussed here.

The simplest encoding scenario is if the BAB is fully opaque or fully transparent. In these cases only a short Variable length Code (VLC) header is added to the bitstream. However, for boundary BABs a more involved processing is required. To encode a binary alpha pixel within a boundary BAB, the shape encoder uses neighbouring pixels (referred to as a context map/template) to generate a *context number*. The context number is then used as an index into a look up table. This indexed entry gives the probability that the current alpha pixel is opaque based upon the neighbouring pixels[9]. The retrieved probability is then used to drive a binary arithmetic encoder. All alpha pixels in the boundary BAB are encoded like this before the output of the arithmetic encoder is added to the bitstream. The context is created either from within the same BAP (i.e.

[9]If the pixel is transparent the retrieved probability is inverted

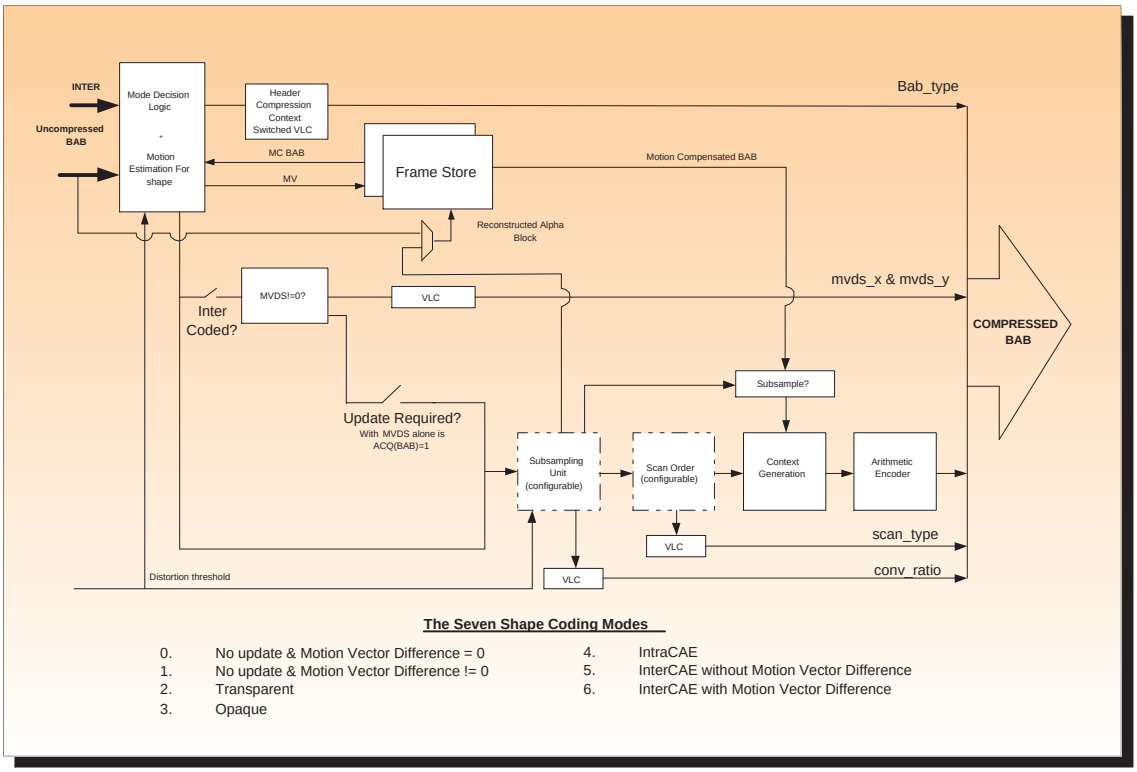


Figure 2.10: MPEG-4 Binary Shape Encoder

intra coding) or a combination of the current BAP and a reference BAP (i.e. inter coding). In total there are seven distinct modes with which a binary shape encoder (see Fig. 2.10) can choose to optimally encode a BAB. The entire process is lossless, although lossy compression is supported through an optional BAB subsampling scheme. An in-depth review of binary shape coding is presented by Brady [44].

In three of the encoding modes when the BAB belongs to an Inter-VOP, temporal redundancy can be exploited through the use of motion estimation. The binary motion estimation process is worthy of further explanation, as it has been shown that it consumes over 90% of the total resources required for binary shape encoding [45]. As shown in Fig. 2.11 a Motion Vector Predictor for Shape (MVPS) is firstly found by examining neighbouring shape and texture macroblocks (16×16 blocks), (see 2.11). The first valid motion vector in the sequence [MVS1, MVS2, MVS3, MV1, MV2, MV3] is chosen as the predictor (where MVS is the motion vector for shape and MV is the motion vector associated with the texture). Once the MVPS motion compensated macroblock is retrieved it is compared against the current macroblock. If the error between each 4x4 subblock of the MVPS motion compensated BAB and the current BAB is less than a specified threshold, the motion vector predictor can be used directly. As the shape decoder uses the same routine to select a motion vector predictor, it will select exactly the same predictor. Thus there is no need to send

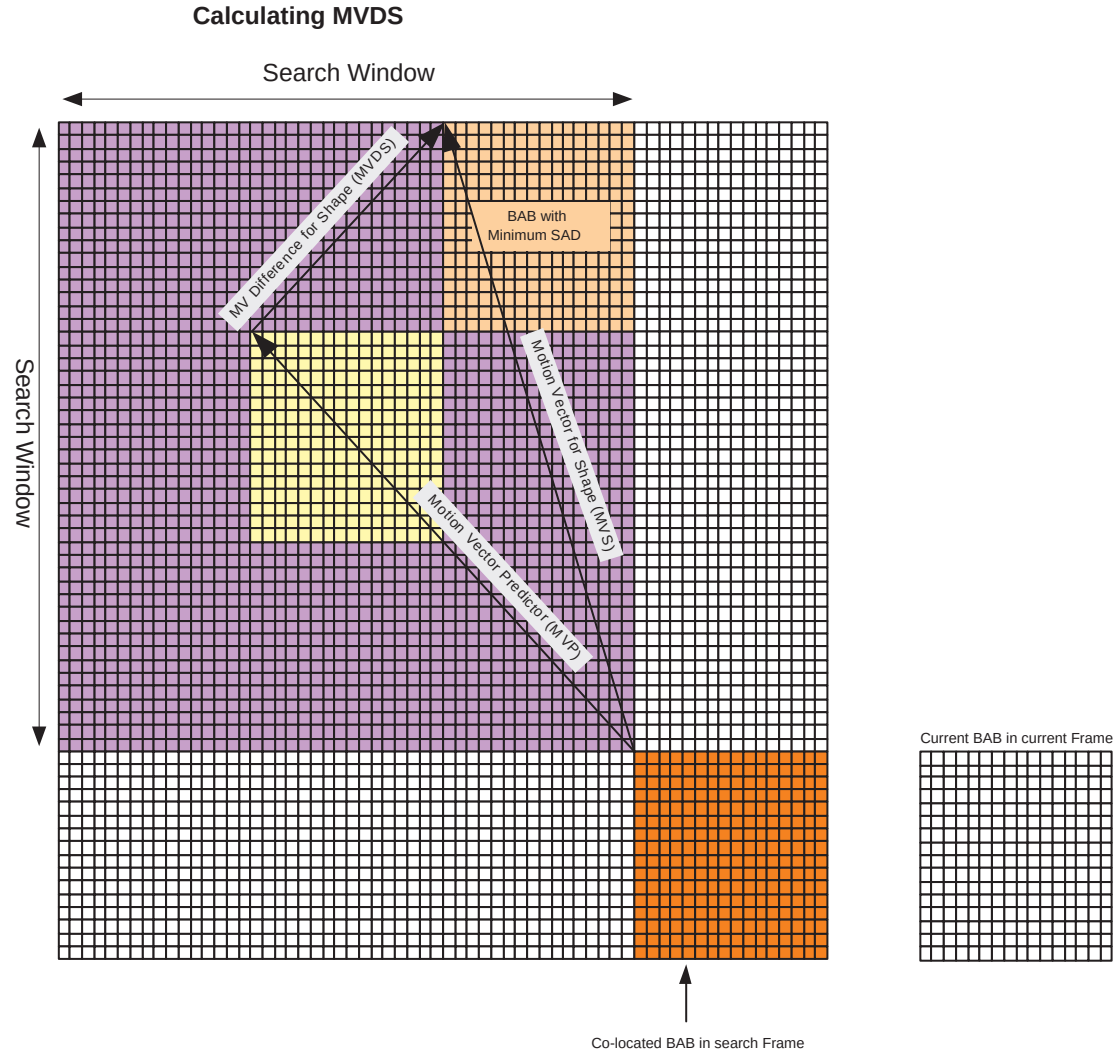
the MVPS to the decoder, reducing the bandwidth overhead. If the MVPS motion compensated error is not less than the threshold a Motion Vector for Shape (MVS) is required. The search window is typically ± 16 pixels around the MVPS BAB. Any search strategy may be used to find the best match within the search window and sub-pel search strategies are not necessary due to the lack of granularity of the alpha information. The designer is also free to choose any distortion metric, with the binary SAD (see Eqn 2.4) being the obvious choice. The bottleneck in binary shape coding for semantic objects within MPEG-4 is the calculation of the MVS. Chapter 7 of this thesis addresses this issue with a dedicated hardware architecture.

2.1.3 Image & Video Compression Standards

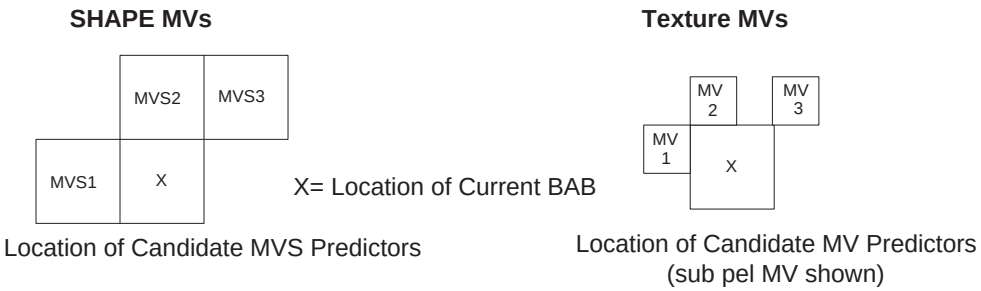
Standardisation plays a pivotal role in the success of modern image and video compression codecs. With the plethora of manufacturers and the many possible platforms to support, codec standardisation ensures interoperability. This can also help to accelerate consumer acceptance of a particular standard. Over the last two decades, groups within the International Standardisation Organisation (ISO) and the International Telecommunications Union (ITU-T) have been the driving force behind video codec standards. The standardisation process is open to all willing participants, with contributions made by both academic and industrial partners. This is facilitated through delegate meetings every two to three months throughout the world, where technical details and time-frames for deliverables are agreed. The following sections give a brief overview of the video codecs that these groups have produced. For completeness, the JPEG still image codec is also be discussed.

2.1.3.1 JPEG

In the early 1980's, study groups in the ISO Picture Experts Group (PEG) and ITU-T were working on compression schemes for digital still images. The separate groups collaborated and formed the Joint Picture Experts Group or JPEG, which became the name of the new image codec. Baseline JPEG firstly applies the DCT transform to the pixels from the input image (colour conversion and normally chroma subsampling is initially carried out). This is followed by quantisation of the DCT coefficients, zigzag scanning, run length encoding, descriptor generation and finally entropy coding. Extensions to the JPEG standard, include progressive encoding/decoding. With this extension an image can be decoded with progressively greater detail. This is useful for Internet applications, as it allows an image to be viewed (albeit in lower quality) before downloading is complete. A non-standardised JPEG extension catering for video is known as Motion JPEG (MJPEG). This



Finding the Motion Vector Predictor



The first valid vector in the order [MVS1,MVS2,MVS3, MV1,MV2, MV3] is chosen as the predictor

Figure 2.11: Motion Vector difference for Shape

effectively treats each frame as a single picture and encodes it as such. In many respects this could be considered to be similar to an intra only video codec. The latest incarnation of the JPEG standard is known as JPEG 2000. The fundamental difference from the original standard is that the DCT transform is replaced by a wavelet transform. JPEG-2000 achieves greater compression with less objectionable artifacts (due to the properties of the wavelet transform). Additional features of JPEG-2000 include region of interest coding and digital watermarking.

2.1.3.2 H.261 and H.263

The demand for low bit rate video telephony and video conferencing in the mid to late 1980's were the motivating factors for the ITU-H H.261 video standard. The standard principally targeted ISDN networks and as such operates at multiples of 64 kbps with support for CIF and QCIF resolutions. It is a Hybrid Differential Pulse Code Modulation (DPCM)/ DCT codec (see Fig. 2.3(b)), and bears much similarity to JPEG but with integer motion estimation to exploit temporal redundancies.

The goal of the follow on ITU-T standard, H.263, was to improve the coding efficiency, particularly for very low bit rate (sub 30-Kbps) applications such as wireless video conferencing. To achieve this goal, a number of new technical features were added. Half pel resolution motion estimation and multiple block sizes are used to generate a temporal prediction residual with less energy. The half pel resolution is generated using bilinear interpolation. Furthermore, the concept of unrestricted motion compensation was introduced. This pads boundary blocks to allow a search window extend beyond frame boundaries. This can be useful for objects moving in or out of a frame. Blocking artifacts were tackled through the use of overlapped motion compensation.

2.1.3.3 MPEG 1, MPEG-2 and MPEG-4

MPEG-1 was the first video standard from the ISO Moving Picture Experts Group (MPEG). During development the standard had a video and audio compression bitrate target of 1.5 MB/s, which was the rate supported by CD-ROM drives of the time. This was important, as it was envisaged the MPEG-1 would support the Video CD, a format intended to compete with VHS. MPEG-1 used H.261 as a starting point for development and as such is very similar to the generic hybrid DPCM/DCT hybrid structure of Fig. 2.3(b). Although MPEG-1 supports resolutions up to SDTV, it does not support processing of enough frames per second at these resolutions for SDTV applications. In line with the goal of creating a standard to compete with VHS, MPEG-1

only supports SIF/CIF resolutions at 25/30 Frames Per Second (FPS). Furthermore, none of the MPEG-1 tools could explicitly handle interlaced video. Such content needs to be deinterlaced firstly for MPEG-1, however as advanced deinterlacing algorithms (i.e. needing accurate motion estimation) are required to minimise high frequency artifacts, a loss in quality and/or compression rate is inevitable. These two aspects were an issue to support digital TV broadcasting and DVD markets, but were addressed in the subsequent MPEG-2 standard. The standard was actually a collaboration between MPEG and the ITU-T, and is officially called H.262/MPEG-2, although is more frequently just referred to as MPEG-2. The range of supported resolutions and frame rates were greatly extended and had a target bitrates of 4-15 Mb/s. Furthermore, the coding efficiency of MPEG-2 is improved by approximately 50% over MPEG-1. Scalability was also introduced, and this increased flexibility allowing different clients to decode different spatial and/or temporal resolutions. A further novel aspect of MPEG-2 was the use of profiles and levels which aided implementation and interoperability but without compromising the standard.

The aim of MPEG-4 was to further improve compression efficiency and flexibility. Initial development leveraged H.263, but the scope grew considerably to encompass compression and manipulation of a vast array of digital media content. Bit rate targets ranged from very low (suitable for video stream to mobile devices) to very high (studio quality). Potential target applications varied from mobile multimedia, streaming Internet video, networked video games, interactive digital TV , virtual TV studio, studio production etc [46]. To support these applications and the coding of both natural and synthetic content, a variety of new coding tools were introduced, these included object based compression (see Section 2.1.2), 2D/3D mesh coding, animated content coding and sprite coding. Clearly with such a wide scope, not all tools are appropriate for all application scenarios. As a result 19 profiles were introduced with differing supported resolutions and coding tools. The fundamental goal of improving compression efficiency (particularly for low bit rate applications) was achieved through the use of such techniques as quarter pel motion estimation, global motion estimation, unrestricted motion vectors, up to four motion vectors per block and the intra prediction of the DC DCT coefficients. Additional partitioning and reversible VLCs were introduced for transmission robustness and error resilience.

2.1.3.4 MPEG-4 Part 10 / H.264

The most recent international multimedia standard is H.264 / MPEG-4 Part 10 Advanced Video Coding (AVC). As is obvious from the name of the standard, this is a further collaboration be-

tween ISO MPEG and ITU-T. Initial development work used the extensions to H.263 and the MPEG-4 simple profile as a starting point. Though H.264 is much narrower in scope than MPEG-4, it focuses on improved coding efficiency and error resilience for rectangular frames of video and also improves the flexibility for usage in different networks types and application domains. Whilst H.264 retains the DPCM/DCT hybrid codec structure, there are a number of advances in terms of video compression efficiency. In part, progress in the available computational resources as dictated by Moore's Law, allows more computational demanding algorithms to be considered. It is claimed that H.264 improves coding efficiency by 50% over MPEG-2 and it is suggested the coding efficiency gains are due to a collection of many small improvements rather than a fundamental algorithmic shift from prior standards [41]. These evolutionary improvements include variable block size motion estimation, quarter pel motion estimation, multiple references frames for motion estimation, improved intra prediction, smaller blocksize integer based transform, context adaptive VLC (CAVLC) and context adaptive binary arithmetic coding (CABAC). Furthermore, a new deblocking filter contributes to perceptual improvements.

In contrast to MPEG-4 which has 19 profiles, H.264 initially had only 3 profiles. These profiles are *Baseline, Main* and *Extended*. The baseline profile allows inter and intra coding along with CAVLC entropy coding. The main profile has support for interlaced content and allows inter coding with B-slices[10] and uses CABAC. The extended profile does not permit interlaced content or use CABAC but allows efficient switching between different types of slices and improved error resilience. There is considerable flexibility within the profiles, yet taking a somewhat simplistic view of profile target applications domains, the baseline profile is considered appropriate for mobile video type applications, the main profile is suited to HDTV, broadcasting and storage applications, while the extended profile suits streaming media applications [35]. Since initial standardisation, additional profiles were added to H.264, these include High Fidelity Range and Scalability profile extensions. It is also worth noting, that unlike MPEG-4 part 2, H.264 does not have support for handling arbitrarily shaped semantic objects.

2.1.3.5 MPEG-7 and MPEG-21

MPEG-7 and MPEG-21 are not focused on compression, but are rather frameworks for the management of digital media content. For example, MPEG-7 Visual specifies four areas for description: colour, texture, shape and motion. Within each category there are simple and complex

[10] A slice is defined as collection of one or more macroblocks, with minimal interaction between slices to improve error robustness

descriptors [34]. Example descriptors for colour include dominant colour, colour histogram and colour quantisation. These descriptors can then be used for further analysis and/or retrieval applications.

2.2 Semantic Video Object Segmentation

Algorithms which segment semantic objects and regions can be categorised as either fully automatic unsupervised approaches or alternatively as semi-automatic approaches whereby the user can interact and guide the segmentation process. In both categories, spatial analysis is frequently used to merge regions of homogeneous colour [47]. By their very nature, semantic objects often consist of differently coloured regions and as a result colour information alone will typically lead to over segmentation[11]. Motion analysis based approaches have also been proposed [48], however object boundaries are difficult to extract using only motion information. The combination of colour and motion information can also fail when the spatial and/or temporal information is not homogeneous within the object [49][50]. Therefore, other features and tools are frequently used to aid the process, these include edge extraction [51][52], active contours [53], gradient extraction [54], texture filters, change detection, depth information, etc.

It has been suggested that semantics can be decomposed into a hierarchy of 4 general levels: visual (e.g. oval), generic objective (e.g. face), instantiated objective (e.g. Mary's face) and abstract/emotional (e.g. important) [55]. Therefore, even when using a combination of the outlined segmentation techniques, it can still be very difficult to map low level pixels and features into higher level semantics. To capture higher level semantics, spatial and temporal segmentation methods are frequently augmented with domain knowledge assistance. For example, the combination of visual, audio and domain knowledge has had success in detecting abstract semantics (such as finding exciting moments) in constrained domains like sports games and films [56][57].

To identify instances of general semantic object classes (e.g. face, car, etc), typically specific object detectors are required. The objects can prove useful by themselves in generic objective tasks, for example in searching for faces in an image database. Moreover, there are those who actually believe that identifying and combining a reduced set of mid level semantic objects and concepts (e.g. "face", "sky", "beach", "building", "vehicle" etc), represents one of the more promising approaches to bridging the so called *semantic gap* [27][28][29]. This belief is based on the fact

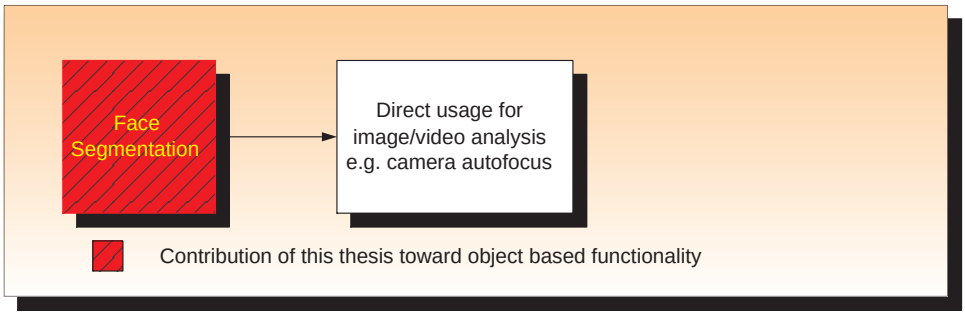
[11] Over segmentation is the detection of excessive number of objects, which in fact may or may not even be consistent an object

that mapping low level pixels to (frequently well defined) semantic objects is a considerably easier task than a direct mapping from pixels to diverse user semantics. In this approach, a possible technique to map the objects to higher level user semantics involves combining and merging the objects through a taxonomy or ontology [29]. As the range of possible objects to detect is vast, for this approach to be viable, it is only practical to detect a smaller subset of domain specific objects.

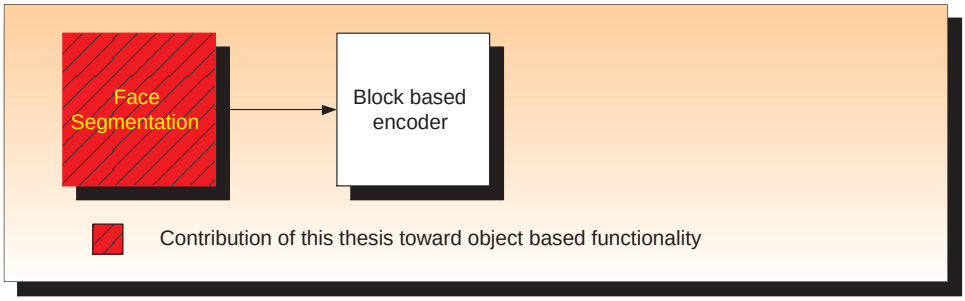
The approach adopted in this research uses the mid level semantic object paradigm. Furthermore, the author believes that detecting objects at the point of creation on a mobile device rather than offline on a less computationally constrained computing resource is an attractive proposition as it allows semantics to be introduced earlier in the digital visual data "life cycle". This has the potential to permit a greater range of novel applications (see Chapter 1). For example, during a mobile video call, the detection of the most important semantic visual objects, would allow higher quality encoding or increased error robustness for these objects. Clearly this is not possible if the object detection does not take place on the mobile device. To constrain the solution space in order to build a robust solution, the presence of a human face in video sequences and images is considered by the author as a fundamental object of interest and particularly to users of mobile devices. The proposed face detection algorithm is described in Chapter 5, and although this method primarily focuses on the detection of "face objects", the approach offers the scope to be retargetted to detect other well defined objects.

2.3 Thesis Contributions in the Context of Video Object Processing

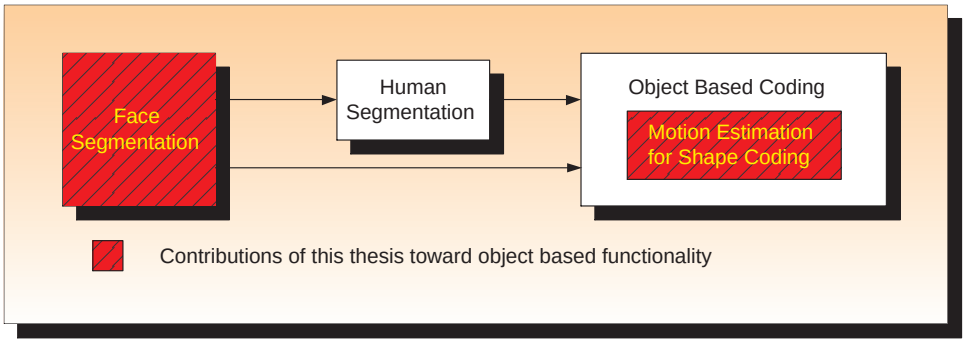
As outlined in Chapter 1 and again in Section 2.2 of this chapter, one of the principal challenges to object based processing is the initial segmentation of the semantic objects. Section 2.2 concluded that the segmentation of mid-level objects offers a promising route to tackle this problem. As a result, this approach has been chosen for further investigation within this thesis. Due to the limited computational resources, battery life and screen size on mobile devices, it is reasonable to further constrain the domain. Consequently, segmentation of the human face is chosen as a primary object of interest to users of mobile devices. Subsequent processing of the segmented face can manifest itself in numerous useful applications for the end user, which in the highly competitive mobile device market, could be considered as vital product differentiators. A direct use of the segmented face on a mobile device is to allow intelligent camera auto-focus. This application scenario (see Fig. 2.12(a)) could be considered to be a highly self-contained use of the face object, since an



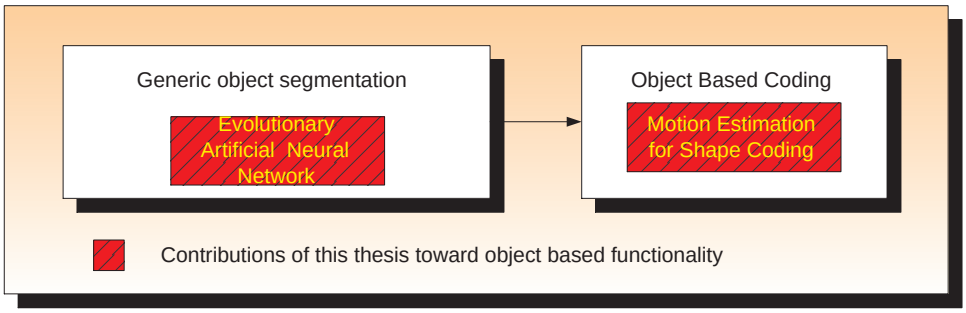
(a) Using the segmented face directly



(b) Using the segmented face in a block based codec



(c) Video object based codec leveraging the segmented face



(d) Video object based codec using generic objects

Figure 2.12: Contributions of this Thesis in context of Video object-based functionality

object based compression platform (e.g. MPEG-4) is not required to process or transmit the face object. The contribution of this thesis toward this type of application scenario is the novel face detection algorithm and associated hardware accelerator that is proposed in Chapter 5 and Chapter 6 respectively. A similar application scenario is shown in Fig. 2.12(b), where the the segmented face is used to increase error protection or ensure higher quality in a semantically important region of an image/frame. Whilst an object based video codec is ideal for this type of application, a non-object based codec such as MPEG-2 or H.264 could also be readily altered to use the segmented face mask to adapt (e.g. the quantisation parameter to improve quality) on a per block basis. The most obvious use of the segmented face is within a video object based compression framework. Although a segmented face can be used by a block based codec, using a video object based codec allows a greater range of applications, such as the ability for interactive services. Another scenario, which is shown in Fig. 2.12(c), is where the face segmentation is used as a preprocessing step for subsequent object segmentation. This is also an example of where a mid level object (i.e. face) can be used in the segmentation process of a another mid to high level object, i.e. a person. Again, the contribution of this thesis for this type of application is the novel face detection algorithm and associated hardware.

When using a video object based video codec, such as MPEG-4, the segmented object must be processed by the SA-DCT and polygon matching based motion estimation. The shape of the segmented face would also require separate encoding in the binary shape encoder. Binary shape coding was shown to be the second most computationally demanding sub-block of MPEG-4 coding [32], and within the shape encoder, binary motion estimation consumes 90% of the resources [45]. As shape coding is not carried out in regular frame based codecs, it represents a completely new computational complexity overhead for a object based video codec such as MPEG-4. This is in contrast to the DCT and motion estimation, which do not suffer a dramatic additional cost for supporting objects. It is reasonable to conclude then, that for a computational constrained device, once a segmentation mask is generated, one of the principal subsequent challenges to adopting MPEG-4 object based processing is the computational cost of shape coding. For the same reasons as offloading the computational complexity hot-spot in the proposed face detection algorithm, the author proposes offloading the computational complexity hot-spot in shape coding, namely binary motion estimation, to a dedicated hardware accelerator. This thesis contribution is shown in Fig. 2.12(c). Details of the proposed binary motion estimation architecture are given in Chapter 7. A further motivating factor for a dedicated binary motion estimation architecture is that it could

be reused for object tracking and possibly in semantic object temporal segmentation algorithms. Though it should be noted, that for the general case when generic video object segmentation and processing is required (as shown in Fig.2.12(d)), the principal contribution of this thesis is the hardware acceleration of the shape coding. Additionally, if the EANN hardware accelerator is retrained, it has the potential to classify other input features. This could be leveraged in a generic object segmentation algorithm.

A key common aspect to each of the example applications mentioned in this Section is a requirement for the processing to be done online on the device itself. In this way, the author believes that the proposed binary motion estimation along with face detection constitute important tools for implementing object based functionality on a mobile device. As a consequence, one can think of the output of this research programme, as the beginnings of a toolkit for low power object based processing on mobile devices.

2.4 Implementation Options

Multimedia processing is inherently computationally expensive even for "simple" algorithms due to the continuous stream of data in high volumes (e.g. real time video). Whilst new and emerging multimedia algorithms (e.g. semantic video compression, video object segmentation etc.) are enabling technologies for a plethora of new applications (see Chapter 1), the cost of manipulating the high volume, high speed video data only serves to compound the computational complexity issue. This creates a situation where careful consideration must be given to the implementation of video processing algorithms, particularly on computational constrained mobile devices. In the broadest sense, multimedia algorithms can be implemented on a dedicated device (ASIC or FPGA) or a programmable device (general purpose CPU, DSP, etc) [58]. For large complex algorithms (e.g. video codecs) this is not a mutually exclusive decision, as hybrid solutions consisting of both dedicated and programmable elements are also possible [58]. The implementation choice of dedicated or programmable solution (or indeed the algorithmic partitioning for hybrid solutions) should consider detailed analysis of the computational complexity (through profiling), the real time constraints, scope for pipelining and parallelism, the memory and bus communications requirements and finally the trade off between power consumption, throughput and silicon area [59]. The conclusions of this design space exploration heavily impact on issues such as time to market, unit cost, battery life and overall weight & size of the final system. All of which are key factors in

the price sensitive mobile consumer electronics market.

A dedicated device implementation, with the options consisting of full custom ASIC, standard cell ASIC and FPGA, are recognised to give better performance in terms of throughput and power consumption compared to programmable implementation options. This can be attributed to the fine grain granularity of algorithmic architecture implementation, allowing precise trade offs between power consumption, area and throughput. However, the disadvantages of a dedicated implementation are generally considered to be a lack of algorithmic adaptiveness, flexibility and the additional development time [58][59]. Tseng et al note that while regular memory addressing and operations are easiest to map to hardware, algorithmic adaptiveness can be handled, but with an increase in development time and possibly a reduction in hardware utilisation [45]. Flexibility in the context of implementation options normally refers to the ease of making changes to a final solution. For example, a flexible solution would allow late changes in a product specification or "in the field" updates for new standards or features [60]. Such flexibility is not possible with an ASIC implementation. However, fast retargetting of FPGA's through a software bit file does allow a certain degree of flexibility with a dedicated solution. FPGA flexibility comes at a cost of reduced throughput and increased power consumption compared to an ASIC solution. In general, there is an inverse relationship between flexibility and performance (throughput and energy efficiency), which is even more pronounced for programmable solutions [45].

For higher levels of flexibility and shorter development cycle, programmable solutions, such as general purpose Complex Instruction Set Computer (CISC) or Reduced Instruction Set Computer (RISC) processors are required. RISC processors are a highly popular option for the varying application demands on mobile devices. But as Pirsch observed, general purpose processors do not efficiently process multimedia data as exploiting the special characteristics of the associated algorithms is not possible, resulting in poor hardware utilisation and excessive clock cycles for frequently occurring, yet simple operations [58][60]. Recent advances such as Intel's MMX/SSE extensions in general purpose processor architectures are attempts to address this by exploiting the inherent parallelism in video and image data using a SIMD datapath [61]. Nevertheless general purpose processors are considered to lack the performance required for complex video processing applications [58][60]. Specialised architectures such as DSPs with a focus on arithmetic performance are required for higher throughput, lower clock frequency processing. However development can be more difficult (e.g. due to issues concerning high level language compilers) than for general purpose processors and complete operating systems typically cannot be run on

DSPs [60][62]. Higher throughput can be achieved using special media processors [60]. These are essentially a DSP or general purpose CPU with additional coprocessors for video and audio processing.

Alternative less mainstream programmable solutions for a mobile device include configurable processors, reconfigurable processors, multicore general purpose CPUs and using Graphics Processing Units (GPU). The instruction set of a configurable processor is customised prior to fabrication and in this way can be optimised (instruction set, cache size, coprocessors etc) for a particular application domain [63]. Reconfigurable processors are similar, but allow run time reconfiguration. Multicore processors are becoming commonplace on desktop and server platforms, and are emerging for mobile devices. Although, compiler technology needs further improvement before fully exploiting the potential of multicore processing. A more exotic implementation solution is the use of a GPU for multimedia processing (although GPU general purpose processing is currently restricted to desktop/server platforms). Due to multiple wide datapaths, GPUs were shown by Cope et al to outperform a CPU, while operating at lower clock frequencies [64]. However, when the relative performance of FPGAs was compared against GPUs, it was found that a FPGA implementation gave a comprehensively better performance for two test algorithms (colour correction and convolution). This was attributed to the flexible pipelining and parallelism possible in the FPGA. Cope et al concluded that for high memory usage applications an FPGA implementation will considerably outperform a GPU implementation.

2.4.1 Summary of Implementation Options

The implementation target for video processing algorithms is a fundamental system level decision, and is a complex trade off between throughput, energy efficiency, flexibility, cost and development time. Semantic processing on a mobile device creates extra challenges due to the issue of additional computational complexity on an already limited embedded (typically RISC) general purpose processor. This section highlighted that general purpose processors trade off performance (both throughput and power consumption) for flexibility and ease of development, although DSP solutions and media processors can give improved performance. Whilst a fully software based solution on a general purpose processor requires less design effort and has flexibility benefits, it is widely acknowledged that a hardware implementation offers a higher throughput and lower power solution [65][45]. This superior throughput and power consumption can be attributed to the dedicated nature of the hardware and possibility for fine grain optimisation of the implemen-

tation. Furthermore, exploiting algorithmic parallelism allows reduced execution time, which can result in a requirement for reduced clock frequency and/or voltage. Whilst not all problems can justify the increased design effort and reduced flexibility of a hardware implementation, it is a particularly attractive solution in scenarios where a software based approach on a programmable platform struggles to meet power and performance requirements for algorithms with regular computations operating on parallel amenable data (e.g. real time video processing). Motivated by this throughput and energy efficiency benefit, the proposed face detection solution, which is discussed in Chapter 3, is designed to leverage a hardware accelerator (described in Chapter 6) for the algorithmic computational complexity hot-spots.

2.5 Energy Efficient Design Principles

The rate of battery discharge in a mobile device governs the operating time of the device between recharges. In addition, the battery also has a direct impact on the overall size and weight of the device. These factors coupled with the overarching impact of power consumption on circuit timing performance, silicon resource usage and electronic component reliability have brought energy efficiency to the forefront of design in recent times. In such a design methodology, the usual goal is to minimise the power consumption within an overall budget of performance, silicon resources and energy. Minimising power consumption involves tackling the mechanisms which cause it. These mechanisms, within the context of the approximate average power (P_{avg}) consumed in a CMOS circuit, are shown in equation 2.8.

$$P_{avg} \approx \underbrace{\underbrace{\underbrace{\alpha C V_{dd}^2 f}_{\textit{Switching Power}} + \underbrace{V_{dd} I_{SC}}_{\textit{Short Circuit Power}}}_{\textit{Dynamic Power}} + \underbrace{V_{dd} \underbrace{(I_{sub} + I_{reverse} + I_{gate})}_{\textit{main leakage currents}}}_{\textit{Static Power}}} \tag{2.8}$$

Where:

- α is the node switching activity
- C is the capacitance of the load and can be attributed to logic and the interconnection routing (wiring)
- V_{dd} is the supply voltage
- f is the frequency of the clock

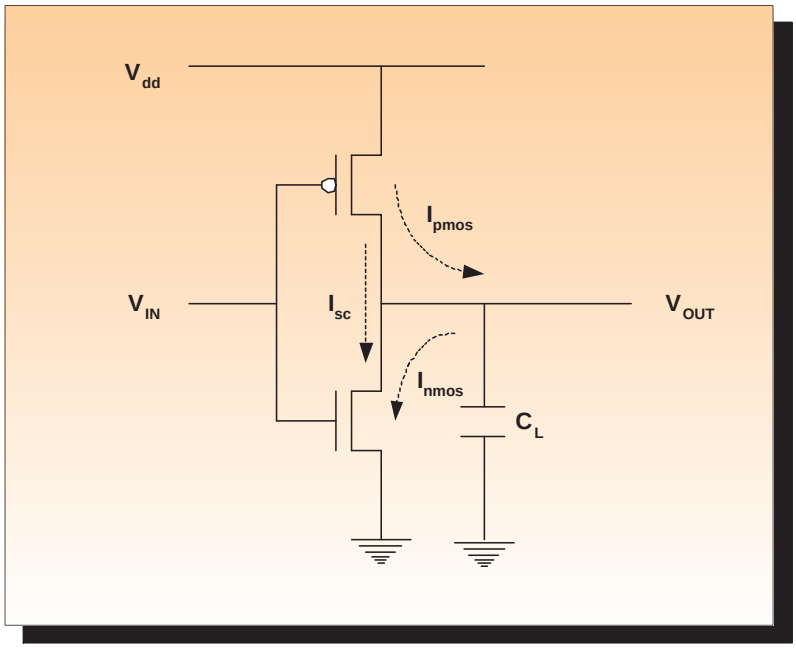


Figure 2.13: Dynamic power loss in a CMOS Inverter

- I_{SC} is the short circuit current, which flows whilst the PMOS & NMOS transistors are both active during switching
- I_{sub} is the the sub-threshold current when a transistor is in an off state
- $I_{reverse}$ is the reverse bias source/drain junction current
- I_{gate} is the gate leakage current

Historically, dynamic power, which is consumed when useful work (i.e. switching) is done in the transistors, was the dominant component in Eqn. 2.8. Dynamic power can be more easily understood via the CMOS inverter shown in Fig 2.13. In Fig. 2.13 the circuit load and parasitic capacitance is modelled as a single capacitor. When the input to the inverter changes from a logic '0' value to a logic '1' value the NMOS transistor has low resistance, whilst the PMOS transistor has high resistance. This allows current to drain towards ground and results in an output of logic '0'. Conversely when the input to the inverter switches back to a logic '0' value, the NMOS transistors has a high resistance, while the PMOS transistor has a low resistance, which causes the voltage source to be transferred to the output and gives a logic '1' output. The power consumed by the CMOS inverter can be modelled as [66]:

$$P_{dynamic} = \frac{dE}{dt} = V_{dd} \times I_{dd}\,(t) \tag{2.9}$$

Assuming a step input voltage and negligible leakage current

$$I_{dd}(t) = C_L \frac{dV_{out}}{dt} \tag{2.10}$$

$$E = \int_0^t P(t)\,dt = C_L V_{dd} \int_0^{V_{dd}} dV_{OUT} = C_L V_{dd}^2 \tag{2.11}$$

In a synchronous digital system the rate of change of the input is related to the clock frequency (f) and the switching activity (α), consequentially the power consumed in a synchronous digital system with a voltage swing of $0 \rightarrow V_{dd}$ is:

$$P_{dynamic} = \alpha C V_{dd}^2 f \tag{2.12}$$

Eqn. 2.12 is the classical equation for (simplified) dynamic power consumption. However, there are additional dynamic power losses, as shown in Eqn. 2.8. For a brief period during the switching of the input, both the NMOS and PMOS transistors are partially on and this causes a short circuit current to flow from V_{DD} to ground. The associated power loss is termed the *Short Circuit Power consumption*. It is generally accepted that by careful balancing the NMOS and PMOS transistors this figure can be kept to 10-20% of the total dynamic power consumption and for this reason is frequently ignored [66][67][68].

Typically for older semiconductor process technologies (e.g. 180nm and above), the dynamic switching power represented above 90% of the total power consumed. However, in sub 100nm semiconductor process technologies, static power is rapidly increasing. Unlike dynamic power, static power could be considered to be wasted energy since its expenditure is not accompanied by a useful contribution to the functionality/processing within the design. The increase in static power can be attributed principally to the vigorous growth in the I_{gate} and (to a lesser extent) I_{sub} currents in Eqn. 2.8 for each subsequent process node reduction below 100nm [69]. The substrate current is increasing because as semiconductor processes shrink to smaller geometries, the oxide layers become thinner. It is now estimated that in 90nm technologies up to 40% of the total power consumption budget can be attributed to static power [69]. A further important consideration is that static power has an exponential relationship with temperature. This is an important design consideration for applications in the mobile domain which may have packaging & heatsink restrictions. In extreme situations an increased temperature will lead to an increased static current, which further causes increased temperature. In this situation thermal runaway and device failure is

possible [70]. It should be noted that as this research is based upon video processing algorithmic investigation on general purpose processors, standard cell ASIC libraries and FPGAs, there is limited scope to address static power consumption, other than powering down logic when not in use. An in-depth discussion of the underlying semiconductor physics behind static power consumption and methods for reducing its contribution are presented in [66][69].

2.5.1 Low Power design techniques

Minimising both dynamic and static power is possible at all levels of design abstraction. Although it is widely accepted the greatest scope for power optimisation occurs at the higher levels of design abstraction, as subsequent design phases reduce the degrees of freedom [71][67][66]. For example, it is not uncommon for design space exploration at an system/algorithmic level to lead to solutions with a factor of 10 to 20 improvement, whereas optimisations at a circuit/process level are likely to yield improvements of only 10% to 30% [72]. Numerous guidelines and techniques can be employed at all design phases to optimise power consumption in a hardware solution [71][67][66]. Minimising redundant operations, avoiding over-design and exploring trade-offs in the power/area/throughput relationship could be considered the general themes across all levels of design abstraction for power minimisation. In the following sections various low power design techniques are discussed which are appropriate at each design phase.

2.5.1.1 System and Algorithmic level power minimisation techniques

Many power optimisations at the system/algorithmic level are application dependant, nevertheless general guidelines can be applied and are discussed in this section. Due to the quadratic relationship between dynamic switching power and voltage (see Eqn 2.8), voltage reduction is an obvious and common approach to power minimisation. For example, in recent times, the supply voltage for general purpose processors has reduced incrementally from 5 Volts down to 1.2 Volts and below. This represents a power saving of approximately a factor of 17. However, there are bounds on the level to which the supply voltage can be reduced, typically two to three times the threshold voltage (V_t) [66], otherwise the resultant performance of the circuit is degraded too severely. Decreasing V_t increases leakage current, which increases the static power contribution. Thus trade offs are required between speed and power. A partial workaround is to use voltage islands and threshold voltage scaling. In this way, the parts of the design which are timing critical can use a high supply and/or V_t, whereas in regions of the design with plenty of timing slack a low supply voltage and/or

low V_t can be used.

From inspecting Eqn. 2.8 another obvious way of tackling dynamic power is to reduce the clock frequency, which has a linear relationship with power consumption. Historically, it was recommended that for a given supply voltage the lowest clock frequency should be used which allowed the design to meet its throughput constraints. However, as reducing the clock frequency reduces the throughput, the device will be powered on for a longer period than would be necessary if a higher clock frequency was used. This can be an issue with the emergence of static power as a growing concern. In many future applications a more energy efficient approach may be found by using a higher clock frequency in order to finish the processing faster and allow the logic to be powered down, in this case, reducing the static power contribution. By considering the throughput requirements of a device, a more holistic approach to voltage and clock frequency can be employed. In particular, if the throughput requirements are varying, it is possible to dynamically vary the voltage and frequency in response to these requirements. One mainstream commercial example of this is Intel's SpeedStep technology [73]. In the extreme scenario, when no processing is being carried out, the logic can be fully powered down. The degree of "power down" is usually related to how fast the logic needs to "wake up" and whether intermediate data needs to be stored. It is clear though from Eqn. 2.8, that reducing the voltage and/or clock frequency will lead to considerable power savings.

A fundamental system level task regularly encountered in System on a Chip (SoC) designs, which can have a major influence on power consumption, is the partitioning of the final solution into software and hardware subsystems. Although a fully software based solution requires less design effort and has flexibility benefits, it is widely acknowledged that a hardware implementation offers a lower power solution [65][45]. This superior power consumption can obviously be attributed to the dedicated nature of the hardware, which effectively reduces the switching activity in Eqn. 2.8. However, if a processor is required in a system, the instruction set architecture plays a vital role in determining the overall power consumption. The options are principally a RISC or CISC architecture. For low power applications typically a RISC architecture such as an ARM core is chosen [74]. In contrast to a CISC core, a RISC core offers simpler instructions with less complex instruction decoding and less switching on the instruction bus. ARM processors also offer a reduced instruction wordlength mode known as THUMB mode, which can further reduce the power consumption, but at cost of possibly more instructions. Another vital techniques for a low power processor is dynamic voltage and/or frequency scaling. Furthermore, reducing the amount

of off-chip memory accessing can lead to significant power savings, since considerable switching is required in the decoding of the address, and the associated activity on the memory and data buses. Memory access reduction is a technique that is used for both dedicated and programmable solutions. In a microprocessor environment a cache is frequently used to reduce the number of off-chip memory accesses, improving both throughput and power consumption. A more recent trend in microprocessor architectures is a shift from single core to multiple core architectures, with each core having a much lower frequency than if a single core was used to achieve to same throughput [61][75][74]. Another technique for dealing with memory power consumption is to use banks of smaller memories where possible since the memory power consumption does not scale linearly.

At the algorithmic level, it is sometimes possible to reduce the word size of the datapath for mathematical operations, thereby sacrificing accuracy/precision of final result for less switching activity (lower dynamic power) and less complex logic (reduced area & static power). In addition, it may also be possible to terminate the computation earlier than normal if some condition exists, saving further processing cycles and power consumption. A classical example of this can be observed in some multiplication circuitry, which terminates the processing early if the multiplicand has a number of leading zeros, as these will not contribute to the final answer. One potential drawback of the technique is that early termination opportunities might only present themselves if a more serial approach is adopted and this may cause difficulties with the throughput requirements. In this thesis, the proposed binary motion estimation architecture (see Chapter 7) uses early termination of block level SAD calculations, whilst also exploiting data-parallelism to improve the throughput. Finally, whilst the obvious choice for data representation in video processing algorithms is regular binary or twos complement notation, it may also be beneficial to explore using different numbering systems. These potentially may exhibit attractive properties such as fewer bit toggles, more lower range precision, etc [76].

2.5.1.2 RTL power minimisation techniques

At the RTL design capture phase, power is principally minimised by reducing switching activity, minimising area and via the use of clock-gating strategies. This section outlines these techniques in more detail. It should also be noted that for some of the more straightforward techniques mentioned (e.g. common sub-expression factoring) modern synthesis tools are also capable of automatically identifying the scope for low power circuitry. Nevertheless, it is frequently desirable for the designer to capture the RTL in a manner such that these optimisations occur.

Although yielding less dramatic power savings, opportunities still exist to reduce switching activity in both the data path and control path even when the overall algorithm details are fixed. For example, by carefully choosing an appropriate state encoding scheme (e.g. graycoding, one hot), control path switching can be minimised in a Finite State Machine (FSM). This is particularly true for a FSM with many states. A fundamental step in reducing data path switching is in the careful selection of the architecture for arithmetic components (adders, multipliers, division etc). For example, a multiple clock cycle booth multiplier is likely to meet tight timing constraints as well as providing opportunities for early termination. Another frequently used technique is the sharing of common sub-expressions. By factoring out common sub-expressions fewer operations need to be completed. This is beneficial from a power and area perspective, although it will create nets with a higher fanout. The use of *don't care* conditions can also be employed to simplify equations/algorithms and thus reduce switching activity. Another example of where switching can be reduced is input operand isolation/gating to parallel logic clouds in which a single result is multiplexed from the multiple logic paths. Glitching minimisation techniques are also used to reduce dynamic power losses. For example, unbalanced logic paths create additional glitching, whereas if the logic path was balanced it will reduce dynamic power as well as potentially improving timing.

Pipelining and parallelism are ubiquitous techniques in hardware design to improve throughput. Moreover, these techniques are also beneficial for power consumption. The throughput improvements due to the introduction of additional pipelining/parallelism can be used to reduce the clock frequency and/or voltage. However, as exploiting parallelism and using additional pipeline stages requires increased area, the balance between area and power requires careful investigation. The flip flops in the register stages (or in filter taps) of a design are a major contributor to dynamic power consumption. A fundamental reason for this is that even when the inputs to a flip flop do not change, power is still consumed due to the internal master-slave latch structure. This can reduced by as much as 30% by *clock gating* [72]. Clock gating as the name suggests, disables the clock input to a flip flop typically via a simple logic gate or latch. The clock is gated during periods of inactivity and the gating signal is generated at a higher level. An alternative approach (although it can also be used in a complimentary fashion) to disabling the clock at the flip flop is to disable a branch of a clock tree.

2.5.1.3 Circuit and Process level power minimisation techniques

From a power consumption perspective, the principal goal at the low level circuit or process implementation phase is to minimise the capacitance in the circuit. In this way dynamic power consumption can be reduced (see Eqn. 2.8). However, it is generally recognised that there is much less scope to reduce dynamic power - reduction figures of 10% to 30% are typically quoted [72]. However circuit level optimisation could be considered to facilitate system/algorithmic techniques such as dynamic voltage scaling (e.g through the use of voltage level shifters). A primary way of reducing the capacitance of the circuit is through the selection of the technology library during synthesis. For energy efficiency a library characterised for low power operation is typically selected. Although, this may impact timing closure, in which case transistors with stronger drive strengths must be selected. During layout, the IO pin positioning on the die will effect the internal routing and could consequently increase wire losses. Minimising this is beneficial for power and timing closure, however there may be a requirement to maintain backward compatibility with previous generations of the chip.

As noted previously, V_{dd} voltage can be reduced to two to three times V_t, whilst still preserving adequate speed in the circuit. Therefore lowering V_t is very attractive from a low power perspective since it allows the V_{dd} to be reduced. However the drawback with this is that lowering V_t slows the transistor and consequentially negatively effects timing closure. Multiple V_t "islands" on the other hand group the logic into domains, and assign a unique V_t to each domain which is appropriate in order to achieve timing closure. Finally, the circuit/process implementation phase is also the most appropriate place to tackle leakage currents and thus minimise static power. This can be achieved using techniques such as the transistor sizing, adjusting gate oxide thickness, usage of high K dielectrics, Silicon on Insulator (SOI), multiple voltage threshold islands and MV-CMOS technologies [65][69][72].

2.5.2 Summary of Low Power Design

This section discussed the sources of power consumption along with typical ways of addressing the mechanisms which cause it. As the target of this work is mobile devices, energy efficiency is a key requirement. It was concluded that the greatest power savings are at higher levels of abstraction. As a result, this research will focus primarily on algorithmic and design capture level (RTL coding) power optimisations. This provides the greatest opportunity for power savings, whilst still being appropriate for a single block whose ultimate deployment is part of a complete system. This is

reflected in the choice of the face detection algorithm, associated hardware architecture and the binary motion estimation hardware architecture. The particular power optimisation techniques used, will be described during the discussion of proposed hardware architecture in Chapter 6 and Chapter 7.

2.6 Conclusions

This chapter presented a comprehensive review of the relevant technical background in order to provide a context and motivation for the subsequent research reported in this thesis. Section 2.1 reviewed digital image and video compression theory, which is a fundamental facilitator for the phenomenal growth in digital multimedia and multimedia equipped mobile devices as was alluded to in Chapter 1. This review of compression theory, included an overview of object based video compression in Section 2.1.2. This section concluded that with the existence of MPEG-4, a complete frame work for object based compression and transport is available. The benefits of object based processing are undeniable, having the potential to enable a variety of new applications. An overview of video object segmentation algorithms was presented in Section 2.2. Following the review in Section 2.1 and Section 2.2, the contributions of this thesis were more clearly defined in Section 2.3 in terms of different types of video object applications. Section 2.4 discussed video processing system level implementation options, whilst the final section in this Chapter discussed energy efficient design principals. It was concluded from these two final sections that dedicated hardware implementations offer a higher throughput and lower power solution than a software implementation and that the greatest opportunity for power savings occur at higher levels of design abstraction. These observations provide the motivation for many of the design decisions made later throughout the thesis.

CHAPTER 3

Face Detection: A Review of Popular Approaches

Face detection in images and video sequences is an essential step in intelligent human computer interaction. Robust face detection can be considered an assisting or enabling technology for a plethora of applications, including object based video coding, video indexing, event detection, emotion awareness, people tracking, etc. When viewed within this wider context, the applications mentioned can be considered as instances of the more general semantic object processing class of applications, where the object of interest in these scenarios is the human face. As such, face detection can be considered a vital component in a semantic object processing "toolkit". With a relentless demand for mobile multimedia (see Chapter 1), it also seems reasonable to expect that robust face detection processing as part of a such a "toolkit", will become commonplace on next generation mobile devices. In fact, this trend is already emerging. Automatic face detection, albeit constrained to single images with a fixed maximum number of faces detected, has recently becoming a highly sought after feature in premium-priced consumer digital cameras [77][78][79]. In this instance, face detection is used to automatically adjust lens focusing and camera settings. This frequently results in considerable visual improvements to photographs where people are present. Furthermore, using face detection in such a scenario also provides an opportunity for automatic semantic markup (e.g. indicating the number of people present, etc.), which could be used later in semantic image retrieval applications. As described in Section 1.2.1, there are many more novel applications which could leverage robust face detection to enrich a user's mobile multimedia experience.

However, before face detection processing gains widespread adoption on mobile devices, a number of technical issues need to be addressed. A key challenge facing designers in this regard, is to find a face detection algorithm, which will give good performance within the constraint of the limited processing capabilities available on the device. Additionally, the algorithm should be energy efficient to maximise the short battery life. The fundamental problem with these requirements is that despite being a well researched topic, face detection remains a computationally challenging task [80][81]. There are a variety of technical issues which contribute to the complexity of the problem. These include environmental conditions (illumination issues, complex backgrounds, etc.), scale variations, orientation variations (upright, rotated, side profile, etc.), vastly differing facial expressions which contort the physical facial structure, occlusion and presence of objects (glasses, beards, etc) [80]. Furthermore, on a mobile device, the associated computational complexity is highly undesirable for both real time operation and energy efficiency. For example, the current state of the art face detection algorithms typically have processing times ranging from 0.055 seconds to 4 seconds for a single CIF sized image when using a desktop processor [82][83][84]. Whilst 0.055 seconds equates to a face detection rate of 18 CIF-sized frames per second, such performance would not be possible on a computationally resource constrained mobile device[1]. This significantly challenges mobile device applications that require real-time face detection in live video sequences. In addition, for many of the applications previously mentioned, face detection is only one tool in an overall semantic object processing algorithm. Therefore face detection impacts the proportion of the real-time budget available for subsequent processing.

One possible solution to address the computational complexity is to offload the face detection processing from the host mobile processor to dedicated hardware, thereby reducing the overall system power consumption (as discussed in Chapter 2) and improving the throughput. However, porting existing software based algorithms to hardware may not necessarily lead to an efficient usage of the dedicated hardware resources. For example, as Section 3.1.3 will show, there are approaches proposed for face detection processing in hardware, however the face detection rates of these approaches have suffered due to over constraining the problem, the choice of algorithm and hardware porting issues arising from modifying an existing algorithm. These observations provide the motivation to design an efficient face detection algorithm, which from the outset is designed with a hardware implementation in mind. The goal is to offer a good trade off between

[1]To determine the exact performance degradation, features of the embedded processor such as instruction set architecture, cache size, clock cycles per instruction, memory access times, etc., should be considered. However, based solely on a naive comparison of typical current clock frequencies for both platforms, it is reasonable to surmise a performance degradation of at least greater than one order of magnitude.

quality of face detection results and amenability for hardware implementation for deployment on a power constrained mobile computing platform. This chapter principally reviews current state of the art face detection algorithms with a view to identifying suitable approaches for hardware implementation. Current hardware implementations of face detection are presented in Section 3.1.3. In Section 3.2, a discussion of the prior art coupled with a description of the constraints and opportunities offered by a dedicated hardware implementation leads to convergence on the proposed approach. Fundamental background theory on the proposed approach is given in Section 3.3, which provides a context for the algorithmic details in Chapter 4 and Chapter 5.

3.1 Face Detection State of the Art Review

Face detection algorithms can be broadly categorised as feature extraction based approaches and appearance based approaches (see Fig. 3.1) [80][81]. Yang et al also describe further categories for less commonly used approaches [80]. These approaches include template matching, which could be considered similar to appearance based approaches, except that the template is defined manually rather than through a learning algorithm. Furthermore, elements from both approaches are frequently combined in the pursuit of a final robust solution. Feature extraction based approaches typically use a combination of facial features as cues to determine whether a region contains a face(s). Appearance-based approaches could be considered to tackle the face detection problem by predominately using statistical analysis and machine learning methods. The prior art in these two approaches are described in Section 3.1.1 and Section 3.1.2. Section 3.1.3 reviews current VLSI implementations of state of the art face detection algorithms. These sections provide a context for the discussion of the author's preferred approach in Section 3.2.

3.1.1 Feature Extraction based face detection approaches

Feature extraction based methods use facial features as cues to determine whether a region contains a face(s). A generic feature extraction based model representative of prior research in this area is shown in Fig. 3.2. This generic model consists of three principal steps: feature detection, feature combination and face verification. With the possible exception of approaches which only use a single feature and have simpler implementations, these steps can be identified in most feature extraction based face detection algorithms. Once the features are detected they are typically clustered/combined. If multiple features are successfully combined a face verification stage is usually

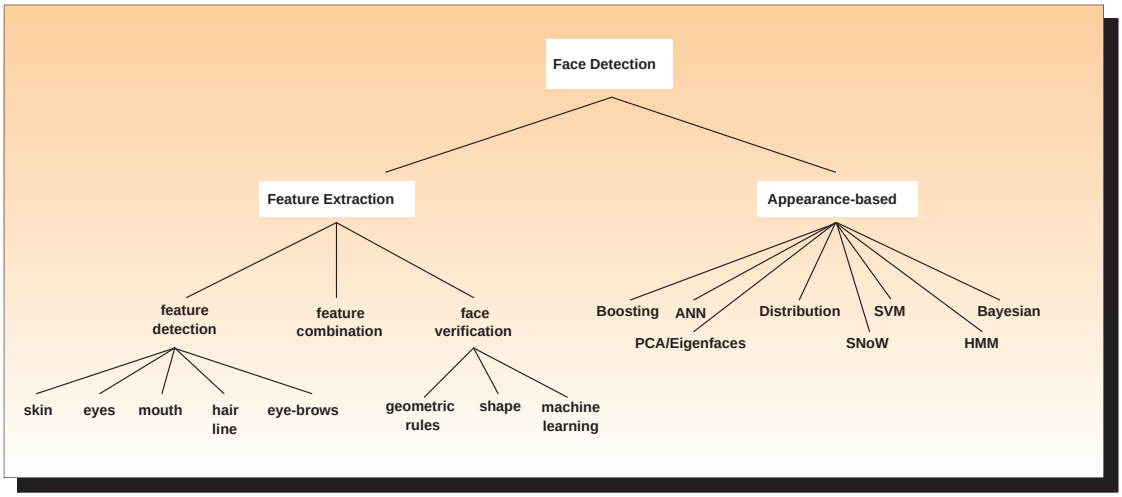


Figure 3.1: Basic taxonomy of face detection algorithms

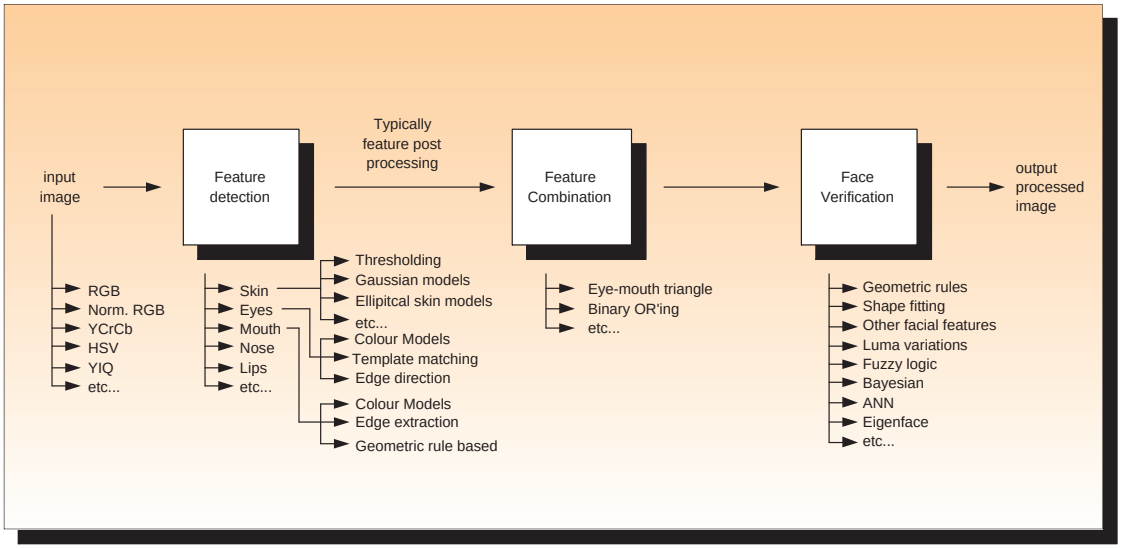


Figure 3.2: Block diagram of a generic feature extraction based face detection algorithm

employed. These three steps are described in more detail in Section 3.1.1.1, Section 3.1.1.2 and Section 3.1.1.3. Feature extraction based approaches have the advantage that features are commonly detected using scale invariant properties such as colour. This avoids searching across multiple resolutions as is frequently the case with appearance based face detection frameworks. The resultant reduced search is advantageous from a computational complexity perspective. However, it can be difficult to extract features in the presence of illumination changes, noise and occlusion [80].

3.1.1.1 Feature Detection

There are numerous structural features of the human face which can be used in a feature extraction based face detection algorithm. These include skin, eyes, eyebrows, hairline, nose, mouth etc. Typically feature detection is the first phase in a feature extraction based face detection algorithm, though in some prior research, pre-processing of the input image is also carried out. For example, Hsu et al use simple rescaling to correct difficult illumination conditions (e.g. shadows etc.) [85] and Ikeda uses change detection pre-processing to reduce the search space in a video face detection application [86]. Depending on the orientation, pose and if there are any objects present on the face, with the exception of skin, not all the features may be present at a given instant. For this reason and the fact that skin detection algorithms are computationally efficient, detection of skin pixels is commonly used in both feature-based and appearance-based face detection algorithms. Regardless of the ethnic origin of a person, it has been found that the chrominance values of skin is constrained to a narrow region in the colour space. As such there is a considerable body of research pertaining to skin detection, which explores a wide variety of colour spaces and classification techniques [87][88]. Commonly used colour spaces include RGB [89][90], normalised RGB [91][92], YCrCb [93][94][95][96][97][98][85][99], HSV [86], YIQ, etc. Detection methods within these colour spaces include thresholding [91][96], statistical modelling [99], elliptical skin models [85][98], Gaussian model [100][92][97], clustering models such as Recursive Shortest Spanning Tree (RSST) [101] etc. Simple CrCb thresholding assumes there is relationship between luminance and skin pixels, however many authors have shown that there is in fact a complex non-linear relationship, particularly when luminance has very low and high values [99][85]. Phung et al use three Gaussian clusters that correspond to low medium and high luminance [99]. Hsu uses elliptical transformations to approximate the effects [85]. With all skin detection algorithms there will be a certain amount of non-skin pixels incorrectly labelled as skin pixels. Frequently these incorrect classifications occur in small isolated pixel groups, which can be removed using techniques such as morphological filtering [94].

Rarely, even under ideal conditions, will skin detection in isolation give an accurate face detection result. Therefore other features are typically used. The detection of eyes, both as a cue for possible candidate face locations and for face verification is common in the literature [85][98][102][95]. A variety of techniques have been proposed for eye detection, including colour eye map models [85], intensity variance coupled with geometric and structural rules [102], thresholded edge extraction [95], Prinicipal Component Analysis (PCA) edge direction [98], morpholog-

ical operator based [103] and template based [104], etc. Furthermore, oscillatory eye movements have also been used as a feature in video sequences [105][106][107]. Similar to the techniques used for eye detection, the mouth/lips can be located via colour models which detect the redness property of the lips [93][85][102]. Other mouth detection methods include edge density with geometric rules [95]. In prior research, noses have also be been used. They have been detected using a variety of techniques including using geometric rules after locating the eyes, relative brightness/-darkness ratios, edge maps etc. Face models which extract features in a more holistic fashion are also relatively common. Active contours are a more advanced version of a holistic approach [108].

3.1.1.2 Feature Combination

If multiple features are used, they must be combined in some way in order to determine whether a location contains a face. This improves the confidence that a particular area contains a face. Hsu et al combines eyes and mouth features using a eye-mouth triangle [85]. The eye-mouth triangle contains inherent properties which can be exploited in a computationally efficient manner for face verification. For example, geometric rules can be applied to determine if the distance between the eyes and mouth is reasonable based upon the distance of separation of the eyes. Furthermore, the orientation of the triangle can also be used. Connected component analysis or clustering techniques are also frequently used [109].

3.1.1.3 Face Verification

The simplest way to verify a face from a group of detected features is to use geometric rules which describe the structure of the human face. Examples include, the proportions of the face, position of eyes relative to nose, circularity [94][99], aspect ratio [99], symmetry about the nose, distance of face from bounding edge of face [85], etc. The geometric rules can be applied by themselves or in a combined manner as a verification step for a candidate face region, which has been generated from clustering lower level features. Other verification techniques used include luma variations of eye and mouth blobs [85], template matching [94], orientation of the eye-mouth triangles [85], elliptic curve fitting [85][99], using other features themselves as verification [93], horizontal edge checking [95], Eigenfaces, PCA [90], fuzzy theory [110], ANN [91], etc.

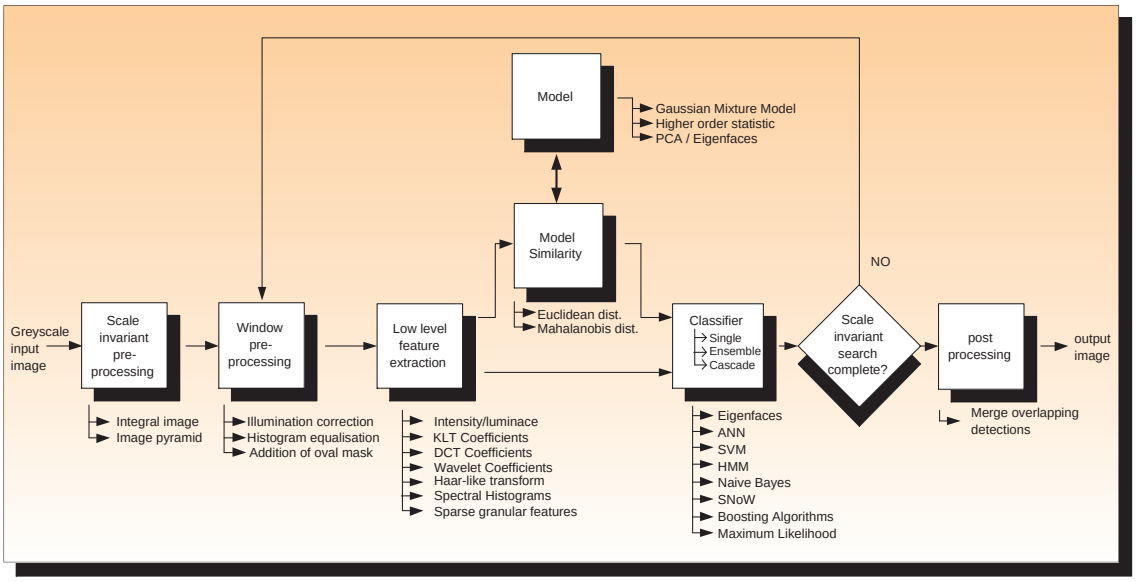


Figure 3.3: Block diagram of the algorithmic options for appearance based face detection approaches

3.1.2 Appearance based face detection approaches

Appearance based approaches treat face detection as a pattern recognition problem, which can be tackled using statistical analysis and/or via learning a discriminatory function. Prior research in this area has principally focused on face detection in full frontal views, though the more complex challenge of side profiles combined with multiple viewing angles is addressed in a more comprehensive manner in recent research efforts [111][112]. The major algorithmic options for an appearance based approach can be seen in Fig. 3.3. The input is generally a greyscale image from which low level features are extracted. The most direct low level feature that can be used is the pixel intensity values. More complex features include ratio templates, KLT coefficients, DCT coefficients, Wavelets (with the simplified Haar transform proving popular in recent research [113]), etc. These more complex features can be useful for classifier dimensionality reduction purposes and in some cases can be considered to have better illumination invariance and rotation invariance properties compared to using the pixel intensity values directly. In contrast to the feature extraction algorithms, which were outlined in Section 3.1.1, high level semantic features such as the eyes, nose, mouth etc. are not extracted. This is beneficial in scenarios in which high level feature detection can prove challenging (e.g. difficult conditions caused by noise, occlusion, change of expression, etc.). Such scenarios are likely to cause feature extraction based face detection algorithms to fail. For this reason, many consider appearance based algorithms more robust than feature extraction algorithms. However, frequently this increased robustness comes at a cost

of increased computational cost relative to a feature extraction based algorithm. A fundamental source of this computational cost, is due to the problem being tackled from a pattern recognition perspective which when coupled with a fixed sized spatial classifier, means that to achieve scale invariance, an exhaustive overlapping block search across multiple resolutions is generally required for each image/frame. A broad range of models, classifiers and associated training schemes have been explored for face detection. Appearance based approaches can be further sub-categorised into Eigenfaces methods [114], distribution based approaches [115], ANN approaches [84][116][83], Hidden Markov Model approaches [117][118], Boosting algorithms [113][119][111][112], naive Bayes classifiers [120][121][122] and Support Vector Machine (SVM) [123][124][125]. The principal representative works in each category are now be discussed in greater detail.

3.1.2.1 Eigenfaces

Using the KLT transformation (also known as PCA), Kirby and Sirovich showed that any face can be approximated using a linear combination of a small number of eigenpictures [126]. They found as few as fifty eigenfaces captured 95% of the variance of a training dataset of one hundred 91×51 face images. Turk and Pentland used this principal to detect faces by examining the residual when a sample is projected into the face space [114]. The residual then becomes a measure of "faceness". The Eigenfaces concept is still widely used in face detection and face recognition research [120][127].

3.1.2.2 Distribution based approaches

Sung and Poggio used a distribution based scheme for face detection [115]. This principally consisted of 6 face and 6 non-face distribution models and a Multi-Layered Perceptron (MLP) ANN to classify samples using distance metrics from a sample to each of the face and non-face distribution models. Each model is a multidimensional Gaussian cluster, whose centroid and covariance matrix was found using a modified k-means clustering algorithm. The modified k-means clustering operates on 19×19 pixel training samples, which have firstly undergone illumination correction, histogram equalisation and have an oval mask fitted to eliminate background structure. To detect faces in an arbitrary image, once the extracted 19×19 candidate blocks are extracted and pre-processed (mimicking the pre-processing steps used in the model training), the normalised Mahalanobis distance and normalised Euclidean distance between the candidate and each cluster centroid is evaluated. A MLP ANN with 24 hidden neurons is trained with back propagation

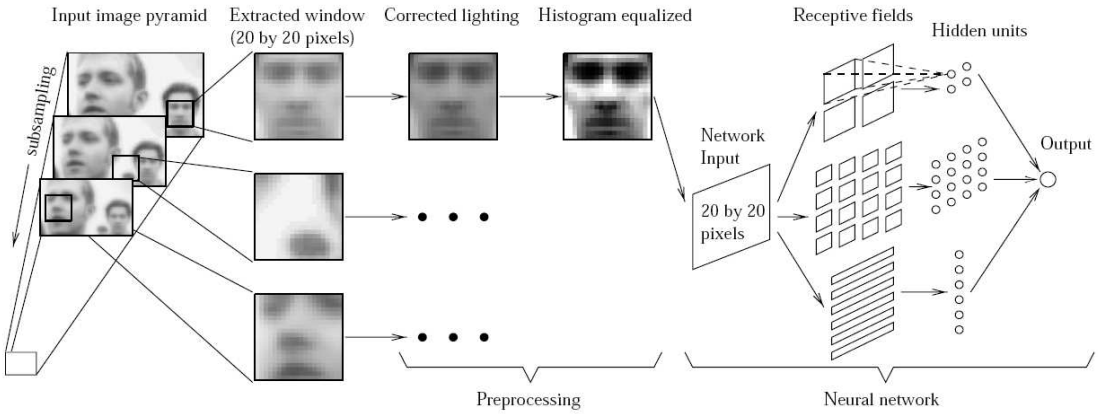


Figure 3.4: Rowley's ANN based face detection algorithm (Source: Rowley [84])

to classify the twelve pairs of distance metrics to give an overall face or non-face decision. To detect faces larger than 19×19, the original image is rescaled in increments of 1.2 to generate what is popularly known in appearance-based face detection algorithms as an image pyramid (see Fig. 3.4). Using the MIT face dataset a recall rate between 80% and 96% was achieved with an acceptably low number of false detections. As well as the overall face detection algorithm, the work by Sung and Poggio is noteworthy for introducing a number of techniques which were used extensively in subsequent face detection research. These include sample preprocessing for normalisation purposes and the bootstrapping technique used in the generation of representative non-face samples. Rajagopalan et al also use a distribution based approach for face detection, but rather than just model the faces and non-faces with multi-dimensional Gaussian clusters they use higher order moments [128]. They claim improved detection performance as a result.

3.1.2.3 Artificial Neural Networks

The work by Rowley et al is generally considered one of the first advanced uses of ANNs for face detection coupled with a large and difficult test dataset [84]. A MLP ANN trained via back propagation is used. The network is trained to detect faces using input pixel intensities and spatial relationships (see Fig. 3.4). Adopting an approach similar to Sung and Poggio, the input pixels undergo illumination correction and histogram equalisation prior to being used in the ANN [115]. Knowledge of the structure of the human face is implicitly embedded in the topology of the ANN. This is achieved by clustering the input pixels into overlapping regions (i.e. four 10×10 pixel regions, sixteen 5×5 pixel regions, six 20×5 horizontal stripes) to detect localised features. This also has the benefit that the ANN is not fully connected, thus reducing the computational cost

of the additional weighted summations of extra connections. The size and location of the regions were hand selected through trial and error. For the training phase, 1050 faces from the FERET face database were used [129]. The face training dataset was extended by manipulating each sample (i.e. mirrored, multiple rotations, scaling, translation). This generated in excess of 15000 face samples. The bootstrapping approach proposed by Sung and Poggio was used to generate the non-face training samples. Multiple ANNs were trained and these contained either 2905 or 4357 connections with 52 or 78 hidden neurons. As the output results of the ANNs were combined during normal mode operation, to ensure diversity in the ANNs, each was trained using different initial randomised weights. During normal mode operation, each ANN evaluates the candidate face pixel region in an exhaustive search of a multi-resolution image pyramid similar to Sung and Poggio [115]. Following this, if one detection overlaps with another, the detection with the lower output value is removed. When examining overlapping pixel regions there is a high probability of multiple positive detection results in the vicinity of a face. Rowley exploits this characteristic to reduce the false positive rate, which has a lower probability of exhibiting this behaviour. He achieved this by employing a threshold for the minimum number of overlapping faces detected in a region before it is classified as a positive face result. Finally, the results from the individual ANNs are combined using Boolean logic. Their approach achieved between 77.9% and 90.3% recall with an acceptable false detection rate when using their large and difficult dataset, which has since been used extensively in face detection research. Rowley's approach requires 50 times fewer floating point operations to classify a window compared to Sung and Poggio's approach [115]. To further reduce the computational cost of the algorithm, Rowley used a 30×30 pixel window and an ANN trained with 20×20 centred and off centred (up to 5 pixels either side) faces. The pixel window advanced in increments of ten pixels in both the horizontal and vertical direction. This ANN introduced considerably more false positives, so a more accurate 20×20 classifier then examined the promising locations from the initial 30×30 search. This reduced the processing time to 7.2 seconds for a 320×240 image using a 200 MHz R4400 SGI Indigo 2 machine, but at a cost of a reduction in the recall and precision rates.

Rowley's initial work could only detect faces in full frontal view with small rotations. Later he extended this work to detect full frontal faces in arbitrarily rotated orientations [130]. This method used an ANN to detect the orientation of a sample. Based upon this value the sample was subsequently automatically rotated back to an upright full frontal orientation and applied to an ensemble of classifiers similar to his previous work. Non-face samples returned meaningless

orientation values, but a non-face rotated by an arbitrary amount should still remain a non-face. However, in some instances the rotated non-face appeared similar to a face. This caused a slight reduction in the detection performance.

Feraud et al claim that learning a discriminant function for face detection is troublesome due to the difficulty of acquiring representative non-face samples [116]. Instead they propose a generative model. This uses a Constrained Generative Model (CGM) ANN, which could be considered to be an auto-associate ANN performing a nonlinear PCA on 15×20 pixel regions. However, as this transformation has an high computational cost, they use pre-filtering to improve run-time performance. The pre-filtering improves the run-time performance by up to a factor of 25. In the first stage of pre-filtering, if the input is a video sequence, they use a simple thresholded frame difference motion filter. Following this, if the original input is a colour image/frame, a skin filter is used to further reduce the number of pixels regions to classify. The final stage before the CGM ANN is a back-propagation trained fully connected MLP ANN with 300 input pixel intensity values, 20 hidden neurons and a single output neuron. Overall the approach achieved similar detection performance as to that achieved by Rowley et al [84].

Garcia and Delakis identified the commonly used sample pre-processing techniques (i.e. illumination correction, histogram equalisation etc.) proposed by Sung and Poggio as a computational cost that could be avoided by using a convolutional ANN [83]. Taking pixel intensity values as inputs to the convolutional ANN, the first four layers in the network topology perform convolution and subsampling, whilst the last two sigmoidal based layer perform classification. The training of the convolutional ANN uses 25212 face samples generated from manipulating (i.e. mirroring, rotating, smoothing, edge enhancing, adding random noise) 3702 randomly collected faces from various sources. The non-face bootstrapping approach proposed by Sung and Poggio was used to generate 25087 non-faces for training. Performance during normal mode operation is 4 frames per second with CIF sized images on a 1.6 GHz Pentium-4. Their approach performed well with rotated faces (rotated up to ± 20 degrees in the image plane and turned up to ± 60 degrees) and overall has a detection rate marginally better than Rowley et al with a lower false positive rate [84]. This lower false positive rate could be partially attributed to their detection merging scheme. After a coarse merging of detections across resolutions using size and centroids of detection as merging criterion, a fine search (with a localised image pyramid) is conducted in the vicinity of the coarsely merged detection result.

One generally accepted disadvantage of ANNs is the need for extensive hand tuning of the

network topology [80]. One possible solution to this issue is to use a Genetic Algorithm (GA) to evolve an optimum topology and/or weights. This approach is known by many names including Topology and Weight Evolving Artificial Neural Networks (TWEANN), EANN or Neuroevolution [131][132][133], though from this point forward EANN will be used in this thesis. As well as reducing the requirement for trial and error design exploration for the ANN, the approach is better equipped to avoid local minima and has the scope for finding a minimal topology [131]. In recent work, Wiegand et al combine EANN techniques to evolve a topology and use gradient descent optimisations to adjust the weight values [134]. Starting with an initial topology of 52 neurons, the algorithm uses genetic mutation operators to add/remove neurons, add/remove links, toggle weight values and add/remove receptive fields. The receptive fields are connections to input pixels which form groups similar to those suggested by Rowley et al [84]. Their goal was to improve run-time operation. The evolutionary search achieved this goal, improving the run-time by 50% relative to a reference topology. This was achieved by the automatic pruning of hidden nodes, whilst still retaining the same classification performance. Unfortunately little further discussion is presented by Wiegand et al about the detection performance or whether standard test datasets were used. Montgomery also recently investigated using a number of Evolutionary Algorithm (EA) based systems with the principal goal of evolving a ratio/template based face detection scheme [135], however, his evolved solutions had high false positive rates.

3.1.2.4 Support Vector Machines

Osuna et al proposed a face detection algorithm which uses an SVM classifier [123]. SVMs can be considered a way of training classifiers. The decision surfaces are found by solving a linearly constrained quadratic problem. The general framework used by Osuna et al shares much commonality with the approaches of Sung and Poggio, and Rowley et al [115][84], in that an exhaustive search of overlapping preprocessed 19×19 pixel blocks from an image pyramid are classified. Similar to Sung and Poggio, bootstrapping is used to generate non-face training data [115]. Detection performance is marginal better than Sung and Poggio, and slightly inferior to the results achieved by Rowley et al [115][84]. Romdhani et al explored using a reduced support vector set for a sequential SVM approach to face detection [124]. This system was 30 times faster than Osuna's algorithm. Waring et al combines spectral histograms and SVM in a recent face detection algorithm [125]. The spectral histogram is generated using a combination of 33 filters (i.e. Gabor filters, gradient filters and Lapacian of Gaussian filters). This algorithm achieves

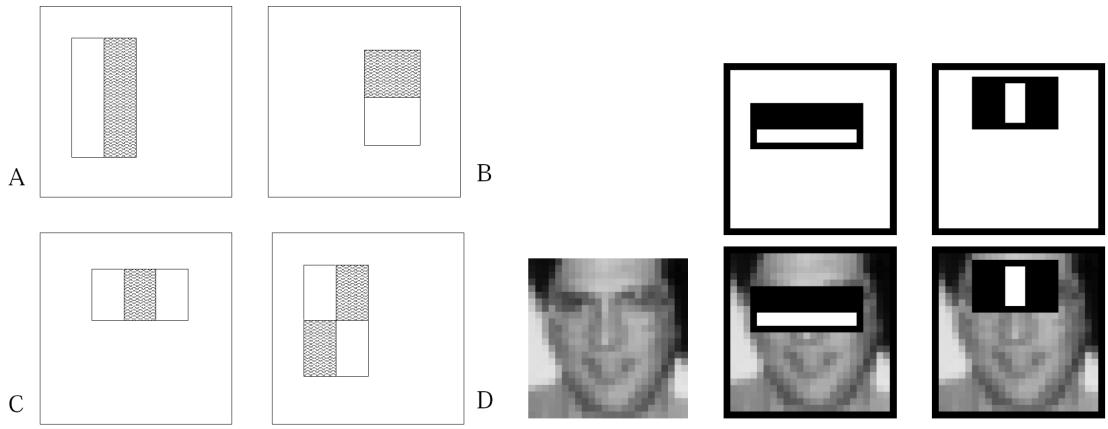
slightly better results than Sung and Poggio, Rowley et al, Yang et al [115][84][136]. However, the cost of evaluating the spectral histogram for each candidate location leads to a solution which requires several minutes to classify a single image.

3.1.2.5 Hidden Markov Models

A Hidden Markov Model (HMM) can be used for face detection by learning the face to non-face and non-face to face transitions through a sequence of observations. Samaria and Young first proposed using a HMM for face detection and recognition [137]. The HMM approach proposed by Rajagopalan et al achieves marginally better detection performance compared to Rowley et al, but with a higher false positive rate [128][84]. Nefian and Hayes combined lower order KLT coefficients as input features with a HMM. This considerably reduced the size of the observation vector and also improved the training and detection run-times. In subsequent work, Nefian and Hayes proposed a similar system, though using a pseudo 2-D HMM and maximum likelihood training [118]. However, comprehensive comparison with other face detection works is omitted by Nefian et al.

3.1.2.6 Bayesian based approaches

Schneiderman and Kanade proposed a face detection scheme where they attempt to model the intractably large and complex posterior probability (i.e. the probability of the presence of an object in image) through a series of constraints, assumptions and learning [120]. Firstly, the size of the object is fixed and this has the consequence that overlapping windows must be examined and also multiple resolutions must be searched for scale invariance. Using Bayes theorems, the posterior probability is decomposed into class conditional probabilities. Four 16×16 pixel subregions are used and a simplification is made that no statistical interdependencies between subregions exist. To reduce the dimensionality of the problem further, a linear projection using PCA is used. The class conditional probabilities are then decomposed into the product of two distributions. Training is used to estimate parameters within the resultant Bayes decision rule. For full frontal images they achieve marginally better detection performance with lower false positive rates compared to Rowley et al [84]. Schneiderman and Kanade later extended this work to more robustly detect side profile faces as well as cars [121]. In this version the statistical information was gathered in the form of multiple histograms of wavelet coefficients. Overall, this approach improved side profile face detection, although full frontal face detection was slightly inferior to their prior PCA



(a) Two, three and four element Haar-like features (Source: Viola & Jones [113])

(b) Two & three element features overlaid on a face training sample (Source: Viola & Jones [113])

Figure 3.5: The Haar like features used in Viola & Jones cascaded Adaboost face detection algorithm

approach [120][121].

Liu also proposed a Bayesian face detection scheme, which integrates feature analysis, modelling and a Bayes classifier [122]. The novel discriminating feature analysis combines the input image, an associated 1D Haar transform of the image and amplitude projections. Liu claims the resultant feature vector improves the discriminatory power for face detection. The face class is modelled as a multivariate normal distribution, as are a subset of non-faces which lie closest to the face class. The model parameters are learnt from training images. After modelling, a Bayes classifier is used to detect faces in unseen images. For full frontal faces, the scheme achieves comparable detection performance to the method proposed by Schneiderman and Kanade but with a lower false positive rate [120].

3.1.2.7 Cascaded Boosting trained detectors

A recent trend in face detection algorithms is to use course to fine grain searches, thus eliminating unlikely candidate locations quickly and focusing computational resources in promising locations. The robust face detection algorithm proposed by Viola and Jones is an example of this strategy and one of the first successful robust real-time face detection algorithms [113]. This algorithm uses cascaded Adaboost trained classifiers. Boosting algorithms can be used with many different classifiers and functions by combining weak classifiers to form strong classifiers. Each classifier in the cascade is trained on the false positives from the classifier in the previous stage. A candidate face sample progresses through the increasingly computationally demanding cascade of classifiers until the sample is classified as a face or until a negative result is found. In this way, processing

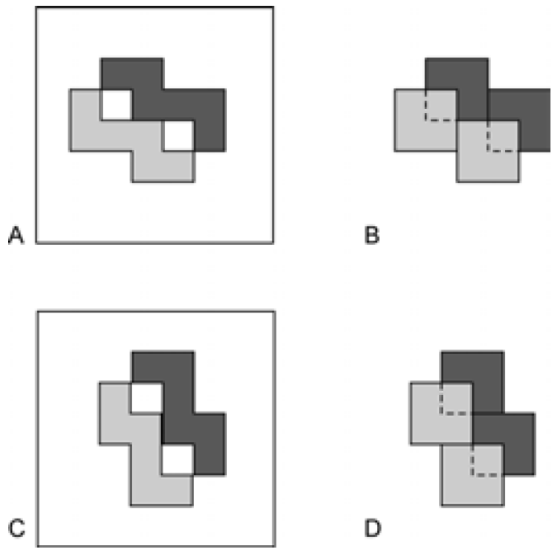


Figure 3.6: Additional Haar features for rotation-invariant face detection (Source: Viola [119])

for obvious non-face samples finishes quickly as the processing terminates in the early stages of the cascade. This dramatically reduces computational run-time and is one of the principal contributions of the approach. To facilitate the increasingly complex classifiers, an increasing number of input features are used within the classifiers. Simple Haar like transformations are used as the input features. These are generated very computationally efficiently using the integral image technique proposed by Viola and Jones. The features, which are shown in Fig. 3.5(a) and overlaid on a training sample in Fig. 3.5(b), are calculated using only simple operations. For example, the two rectangle feature is the difference between the sums of pixel intensities in two rectangular blocks, the three rectangular feature sums the outer blocks and subtracts this from the inner block. The integral image is beneficial as it avoids the computational cost of the image pyramid generation by allowing the detection window to resize. However, for face scale invariance a full exhaustive search across multiple resolutions is still necessary. The final classifier has 32 layers using a total of 4297 features and achieves slightly better detection performance compared to Rowley [84]. However, the run-time is on average 15 times faster than Rowley's method [84]. It achieves 14 frames per second with 320×240 sized images. A variant of Viola and Sung's algorithm was proposed by Šochman [138], which differs in that it uses a faster version of boosting. This eliminates candidates sooner in the cascade and typically improves run-time performance by 20%, although detection performance degrades somewhat.

Viola and Jones later extended their initial system to also allow detection of rotated faces [119]. A decision tree was used to estimate the pose of a sample and a trained cascade of Ad-

aBoost classifiers, similar to their prior research, was used to classify the window. Four new diagonal classifier features (shown in Fig. 3.6) were introduced to improve the detection accuracy of non-upright faces. These diagonal features were carefully chosen allowing for fast generation from the integral image. Again similar performance was achieved compared to Rowley's rotation invariant work [130], however the run-time is considerably better. A side-profile face detector using additional decision trees and detectors gave similar performance to the system proposed by Schneiderman and Kanade, but again the run-time was considerably improved. Wu et al proposed a solution which could be considered to be a variant of Viola and Sung's rotation invariant approach, but used the Real Adaboost algorithm [139][119].

Extensive improvements to the AdaBoost face detection framework were recently proposed by Li and Zhang and more recently by Huang et al [111][112]. Both recent approaches focus on improving multi-view and rotation invariance in addition to improving the boosting algorithm. Li and Zhang use a novel training scheme called FloatBoost, which automatically removes weak classifiers from the cascade that are deemed ineffective [111]. This leads to a better performing strong classifier with an improved run-time. Multiple detectors organised in a course to fine grain pyramid structure are used to detect faces in different views (i.e. from side profile through to full frontal in increments of 10 degrees). Using a full frontal view test dataset (i.e. the CMU dataset) the approach marginally out performs Rowley's ANN approach for a given number of false positives. It also achieves better performance than Viola and Jones original work and uses up to 58% fewer weak classifiers. The multi-view capabilities are further extended by Huang et al [112]. They propose a new cascade structure called width first search, which attempts to tackle the deficiencies in the decision tree approach and the pyramid approach [119][111]. Furthermore a novel generalised boosting algorithm termed Vector Boosting is used in the training phase. Rather than use simple Haar-like features, novel sparse granular features are proposed, which they claim capture irregular features better whilst having similar computational costs. These combined contributions allow the algorithm to outperform the methods proposed by Rowley et al, Viola and Jones and Li and Zhang in terms of detection rate [84][113][111]. Performance is 10 frames per second on a Pentium-4 3-GHz machine when using 320×240 sized images.

3.1.2.8 Other Appearance-based Approaches

Yang et al use the Sparse Network of Winnows (SNoW) training methodology [136]. This is specifically targeted for learning in domains in which there is a large number of features, but only

a small number are active at any time. The input feature used is the rather unusual combination of pixel intensity multiplied by the position. Preprocessing of samples occurs similar to that proposed by Sung and Poggio [115]. The approach is computationally efficient and achieves better results than Sung and Poggio, and Rowley et al [115][84].

3.1.3 Prior art in hardware acceleration of face detection

There are a small number of face detection algorithms implemented in hardware. Etienne-Cummings et al describe a low complexity analog design for Hue Satursation Intensity (HSI) histogram-based skin detection followed by simple user defined template matching [140]. An analog design approach is also used by Boussaid et al, who proposed a threshold-based skin detection filter which uses the normalised Red Green chrominance subspace [141]. A digital based design for skin detection design is proposed by Krips et al, in which a three input (i.e. RGB) ANN is used [142]. The ANN is trained offline with back-propagation. The trained topology is a fully connected feed-forward MLP with three hidden neurons. Owing to the small and fixed topology size, the ANN is implemented directly having dedicated logic for each structural element. The activation function in each neuron is implemented as a Look Up Table (LUT).

Unfortunately, skin detection without further processing is not sufficient for robust face detection (see Section 3.1.1.1). However, a hardware architecture for skin detection followed by such post-processing was proposed by Paschalakis et al [109]. This design, which is targeted for a Field Programmable Gate Array (FPGA), uses a combination of adaptive statistical histogram based skin detection, connected component analysis and a simple tracking algorithm. The input QCIF size frame is firstly subsampled to a very low resolution (11×9). Further processing occurs at this resolution and thus explains the high throughput of the design, which they claim can achieve over 400 frames per seconds using a 33 MHz clock. The skin detection is a threshold-based scheme in the LogRG subspace and is implemented via a series of memory lookups. A simplified connected component analysis algorithm then tags and clusters detected skin pixels. The tracking algorithm uses distance and mass metrics calculated using each skin cluster and the detected face(s) in the previous frame. Unfortunately, there is no evaluation by Paschalakis et al against other approaches using standard test face datasets, thus making comparisons difficult. Furthermore, as this approach is heavily dependent on the initial skin detection process it is unclear how well (if at all) this algorithm will function in presence of skin-like coloured background regions. Non-face skin regions (arms, legs) can also be classified as faces, although there is less opportunity for this to occur in

the proposed application of frontal head and shoulders type mobile video conferencing.

The more common approach for implementations of face detection in hardware is to use appearance based algorithms (see Section 3.1.2). Possible reasons for this design decision will be discussed further in Section 3.2. Implementations of appearance-based face detection algorithms, which include at least some aspect in dedicated hardware, include ANN based approaches [143][144][145], a scheme using a naive Bayes classifier [146], a HMM approach [147], PCA/eigenface based face recognition [127], and a recent implementation using the AdaBoost algorithm [148]. The hardware ANN implementation proposed by Theocharides et al is based upon the rotation invariant face detection ANN algorithm proposed by Rowley et al [130][143]. Theocharides et al implements dedicated hardware for the image pyramid generation, histogram equalisation, illumination correction, rotation detection ANN, and the retinally connected ANN classifier [143]. A word-length of 9 bits (1 sign bit, 3 integers bits, and 5 fractional bits) is used within the ANN elements. The Tanh activation function is implemented in a LUT. The design is synthesised for 0.16μm ASIC technology and achieves a frequency of 75 MHz. However, the detection rates are considerably lower than reported in [130]. This is most likely a consequence of the look up LUT based activation function and/or insufficient precision in the fixed point arithmetic. Furthermore, the power consumption of 7 watts is unsuitable for mobile deployment [143].

A hardware / software ANN implementation is proposed by Sadri et al [144]. Numerous functions including image pyramid generation, rotation invariance and the MLP ANN classifier are processed in dedicated logic on an FPGA. The remaining tasks (e.g. detection merging, etc.) are carried out in software on a PowerPC which is part of the fabric of the FPGA that is used. In contrast to the approach of Rowley et al, after pre-processing a sample, an edge filter is applied, resulting in a binary quantisation of the input pixels. This has the beneficial effect of eliminating the multiplication operations required in the first layer of the ANN topology. Although, it is also likely that the quantised pixels will cause a reduction in the overall face detection performance. However, as the performance of the system is not compared against others or indeed evaluated on a standard test dataset this cannot be verified. Similar to Theocharides et al, a word-length of 9 bits and a LUT based activation function is used. After the first layer in the ANN, embedded multipliers on the FPGA are used for the weighted sum calculations of ANN connections. A throughput of 9 frames (800×600 pixel resolution) per second is claimed with a 200MHz clock frequency. Smach et al also describe a ANN based hardware implementation for face detection [145]. A linear spline based approximation for activation function was used, although this logic

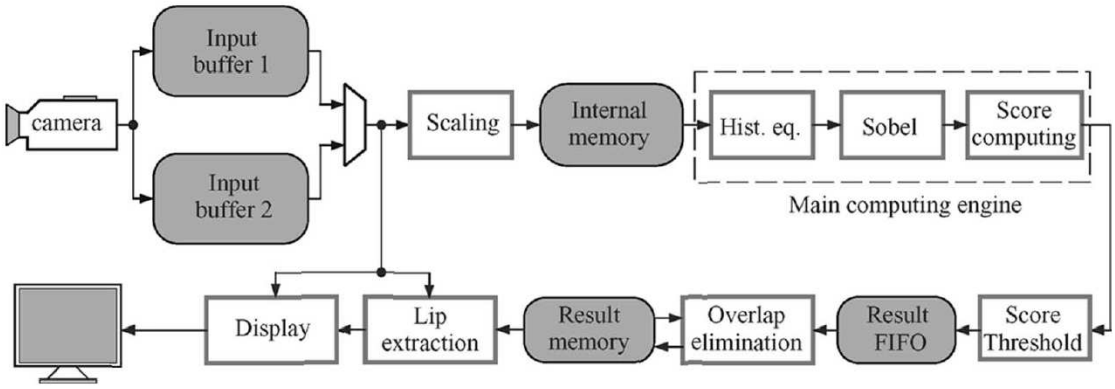


Figure 3.7: System Architecture of Face Detection Algorithm proposed by Nguyen et al (Source: Nguyen [146])

was present in each of the 16 Processing Elements (PE). Had a systolic array architecture been employed, this could be easily avoided. No discussion is presented by Smach et al concerning the face detection performance of the resultant system.

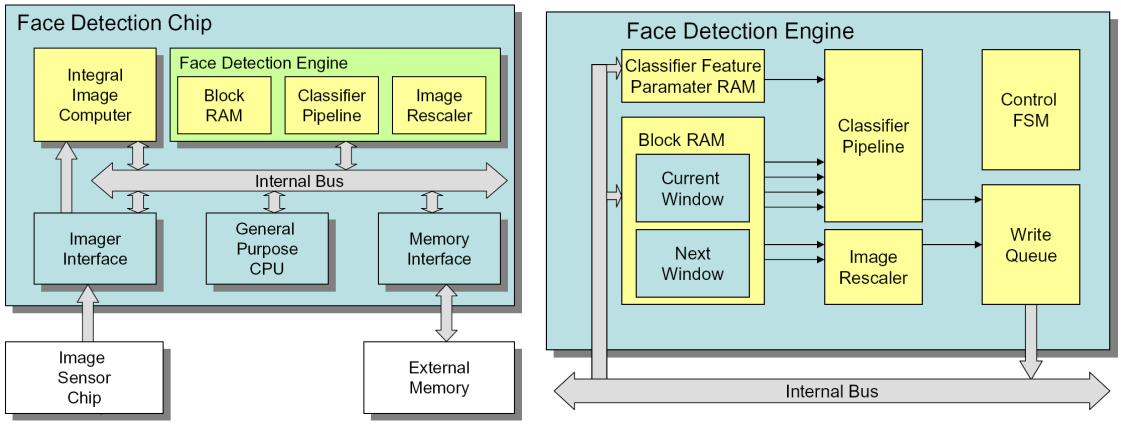
A hardware PCA based face recognition system is proposed by Prasanna et al [127]. An ANN is used to calculate the principal components after training with generalised Hebbian learning. The design is implemented using 0.13μm ASIC technology and achieved a maximum clock frequency of 117.5 MHz. Using the Yale database 75 images were processed in 12 seconds. They claim a speed up in the order of 10^5 is possible when using the dedicated hardware over a PC implementation, although, this comparison is not strictly fair as the PC implementation is running within Matlab, where its execution speed is constrained by the interpreted environment.

Nguyen et al present a face detection scheme which is used in an algorithm to extract lip features for a combined audio visual speech recognition application [146]. Their proposed face detection algorithm combines image gradient information with a naive Bayes classifier. An exhaustive search using 20×20 pixel sized windows over multiple resolutions is used. For each extracted window, histogram equalisation is carried out. The magnitude and direction of the edges present in the window, which are extracted using a Sobel filter, are subsequently calculated. A quantised version (7 levels) of the edge direction is then used to form a vector which is classified using a naive Bayes classifier. A block diagram of the system architecture is shown in Fig. 3.7. The score computing unit, which effectively encapsulates the face model, uses the quantised edge direction vectors to retrieve 16-bit (i.e. using 1.1.14 representation) likelihood values from LUTs in Read Only Memorys (ROM) on the Altera Stratix FPGA. The retrieved values are then summed using an adder tree. This design was targeted for an FPGA, where it used 15,050 logic elements

and achieved a maximum clock frequency of 41 MHz. The algorithm was evaluated using the Yale dataset, where it achieved 86.6% detection with no false positives.

The hardware architecture proposed by Yang et al is based on Viola and Jones' AdaBoost face detection algorithm [148][113]. A number of modifications and approximations are firstly made to Viola and Jones' algorithm for the intended application scenario of this work, whereby face detection would be used for automatic exposure control in handheld cameras. As such, strict real-time constraints must be placed upon the original algorithm. This is troublesome in particular for the variable run-time of the cascaded classifiers. In the extreme case, rather than truncate processing to ensure a specific run-time performance, which they show can lead to a considerable number of missed detections, they propose a scheme exploiting spatio-temporal properties of the cascade and source images. Furthermore, rather than rescale the detection window, they rescale the integral image. They claim adopting this strategy is beneficial because on embedded processors with very small cache size, the cost of rescaling is less than the cost associated with the loss of data in the cache, due to the larger detection window size. This cost differential is especially true as the integral image rescaler can run in parallel and does not require substantial logic. The hardware architecture proposed by Yang et al is shown in Fig. 3.8(a). The face detection engine shown in Fig. 3.8(b) is principally responsible for retrieving the features, computing the classifiers and rescaling the image. The elements of the integral image are stored using a word-length of 32-bits. The classifier pipeline consists of rectangular summation, a four cycle multiplier and a threshold comparison. The latency of the classifier pipeline is five clock cycles, but once filled one feature is processed every two cycles. The design is targeted to an Altera Cyclone II FPGA and uses 6394 logic elements and 22 Kilobytes of storage. The implementation is evaluated using a custom test dataset and achieves over 75% detection with 13 false positives.

A number of other prior works in the face detection literature also have low levels of hardware acceleration. These include a resistive fuse network approach proposed by Nakano [149] and an edge feature extraction VLSI chip used in a 2-D HMM approach by Suzuki and Shibata [147]. An interesting design space exploration for face detection, which is applicable to hardware implementation or multi-core microprocessors, is presented by Chellappa et al [150].



(a) System Architecture (b) Face Detection Engine

Figure 3.8: Face Detection Hardware Architecture Proposed by Yang et al (Source: Yang et al [148])

3.2 Discussion: Suitable Algorithms for a Mobile Device

The goal of this section is to draw conclusions on the state of the art review of face detection algorithms that was presented in Section 3.1 and identify a suitable energy efficient algorithm for deployment on a mobile device. Along with the goal of energy efficiency, the other principal motivating factors in the choice of algorithm are the potential quality of face detection results and the run-time performance. Although, it was established in Chapter 2 that dedicated hardware offers a considerable energy efficiency benefit over a software implementation, a fundamental design decision still remains whether it would be appropriate and cost effective to offload all or some of the face detection processing to dedicated hardware. In general, dedicated hardware brings faster execution, improved energy efficiency and reduced manufacturing costs (provided the number of units exceeds a sufficient volume). However, the cost of these benefits is increased design time (and as a result higher non-reoccurring engineering costs) and a reduction in reconfigurability. For any algorithm these trade-offs should be carefully explored within the context of the intended deployment target. For example, a dedicated hardware block for face detection which could be reconfigured for an artificial intelligence gaming application is potentially very attractive and valuable for a multi-function mobile device, but the benefit is less so for a dedicated video sensor node in a security application. It should also be noted that a dedicated hardware implementation is not necessarily faster compared to software for all algorithmic structures. For example, a single multiplication in software is most probably using a dedicated multiplier in the datapath of a full custom designed microprocessor, which is likely to be at least as fast as the same word-length sized multiplier in the pipeline of a hardware accelerator that was designed using a standard cell

ASIC design flow. The possible benefit for the dedicated "hardware accelerator" in this rather contrived example, is that the datapath width could be tuned to the algorithm to allow reduced bit switching (lowering power consumption) and/or multiple multiplications could be carried out in parallel thus reducing run-time. Unlike a software implementation, the dedicated hardware also has the power consumption advantage of not requiring an instruction to be fetched from memory and decoded.

If a hardware design route is chosen (most likely due to real-time performance issues and/or power consumption issues), consideration should be given to how easy the function is to pipeline and whether opportunities exist to exploit parallelism in the function to achieve a specific performance level. In addition, for image and video processing algorithms, careful consideration must be given to memory access and structuring. Otherwise a dedicated hardware solution could transform an initial compute-bound problem to a memory bandwidth bound problem, which will fail to maximise the potential of the dedicated hardware resources. Furthermore, memory accesses are generally considered very power consumptive due to the address decode logic. With these general considerations in mind, the author concludes that a hardware implementation of inherently computational demanding face detection processing on a mobile device makes sense from an energy efficiency and run-time performance perspective provided the specific face detection algorithm can be easily pipelined, offers scope to exploit parallelism and uses regular memory addressing. A more likely scenario for a given face detection algorithm is that there are computational complexity hotspots within the algorithm which are more suitable for hardware offload compared to implementing the entire algorithm in hardware. The remaining operations can reside in software and use the host processor available on the mobile device. This approach also opens up the possibility that with different software control logic the dedicated hardware accelerator could be reconfigured for other tasks.

The algorithms discussed in Section 3.1 will now be re-evaluated from a hardware offload suitability perspective. With the exception of trivial skin detection filters, few feature extraction based face detection algorithms have been implemented in digital hardware [109]. This may be explained by the fact that in general feature extraction based approaches offer limited opportunity to exploit hardware parallelism, involve complex memory addressing schemes, may require multiple passes and are difficult to efficiently pipeline. In contrast appearance based approaches, have many desirable characteristics, which lend themselves to viable hardware implementation. Moreover, many consider appearance-based algorithms to be more robust compared to feature ex-

traction based methods (see Section 3.1). As can be seen in Fig. 3.3, a trained appearance based approach principally consist of an exhaustive search across multiple resolutions, where each candidate location may undergo normalisation processing before a classification phase[2]. As such, memory addressing is regular and many of the proposed classifiers have already been implemented in hardware. Of the many classifiers reviewed in Section 3.1, ANNs, SVMs, naive Bayes classifiers and Haar-like feature classifiers trained with boosting algorithms have resulted in the best detection performance, and all of which have been shown to have viable hardware implementations [143][146][151][152][148]. Since all these approaches are generally accepted to give similar face detection performance [84][123][120][113], and boosting is a technique which can be applied to most training schemes, other factors should also be considered, particularly from a hardware implementation perspective.

Reuseability of a hardware core is important on a constrained computing platform to avoid relatively infrequently used logic causing excessive silicon area. This becomes even more important in deep sub-micron technology, where static power can contribute up to 40% of the total power consumption in 90 μm semiconductor process technologies [69]. Although, in this scenario, the static power can be addressed by powering down the block. Nevertheless, when using reusability as a design criteria, ANNs have a distinct advantage over the other approaches, due to ANNs more established application base, a result of them having been deployed in a broad spectrum of classification, perception, association and control applications [30]. Furthermore, within the semantic image/video processing domain, ANNs are frequently used for segmentation, automatic labelling and image understanding tasks [153]. As such, an ANN can be considered a highly versatile fundamental tool in a semantic object processing "toolkit". One that may be combined with other lower-level basic video processing blocks tools (e.g. illumination correction, histogram equalisation, etc) to achieve a solution for higher level abstracted semantic tasks.

A difficulty when using ANNs is that considerable manual fine tuning to the topology and associated parameters is required to achieve excellent results [80]. Finding an appropriate network topology and an optimal set of weights represents a difficult multidimensional optimisation problem. Ideally, the topology should be as small as possible, but large enough to accurately fit the training data. Failure to find a suitable configuration will cause poor generalisation ability with unseen data and/or excessive execution time. As was mentioned in Section 3.1.2.3, using EANN algorithms, in which a GA automatically searches for optimum topologies, is one potential

[2] An optional model similarity check is sometimes evaluated prior to classification, such as with the approach proposed by Sung and Poggio [115].

solution. The EANN paradigm will be explained more thoroughly in Section 3.3.3. However, a number of the potential benefits offered by EANNs are now be outlined.

It has been shown that an ANN based face detection system with local receptive fields outperforms a fully connected ANN [154]. Local receptive fields allows the ANN to detect localised features [84]. However, as the size and connections for these receptive fields are determined manually, a EANN approach should be able to evolve at least as good a solution. Additionally, unlike a feed forward ANN, such as that used by Rowley et al [84], a modern EANN framework can evolve any type of topology, potentially with a mixture of forward connections, recurrent connections and looped current connections. Combined, these factors have the potential to allow non-trivial creative topological solutions to evolve via the GA.

EANNs also offer the possibility of finding a minimal topology, which is hugely beneficial since fewer neurons and connections lead to reduced computations. This in turn means higher throughput and lower power consumption. EANN based training also offers benefits from a hardware perspective relative to an ANN trained via on-chip back propagation, as the GA optimisation search should be less sensitive to reduced word length implementation issues. For reusability purposes the ANN evaluation of the EANN can be easily offloaded to dedicated hardware. As was also observed by Reyneri, this approach to ANN hardware design combines the scalability and flexibility of software with the speed and energy efficiency of dedicated hardware [155]. For example, by modifying the software, the same core that accelerates face detection could be reused as an EANN accelerator for advanced AI for computer gaming [31]. In the face detection scenario, EANN based training occurs once. After training, during normal mode face detection operation (e.g. face detection during a mobile video call), the proposed algorithm will leverage the accelerated processing performance of the dedicated core. In the AI application scenario, constant GA based evolution with hardware accelerated phenotype evaluation would occur to alter the AI of computer controlled agents in response to the playing characteristics of the human player [31]. As will be shown in Chapter 5, profiling revealed the computational burden of the ANN evaluation is suitable for hardware offload, whilst the GA, which uses moderate computational resources resides in software.

A recent trend in face detection algorithms, in particular with the Haar feature boosting trained algorithms [113], is to use a cascaded structure of increasingly complex classifiers. This directs computational resources to more promising face locations. During EANN training, the evolutionary process generally creates increasingly more complex topological structures. This is especially

true if the EANN algorithm evolves from a minimum topology rather than an initial starting topology with considerable structure present. The Neuro Evolution of Augmenting Topologies (NEAT) open source library is one such EANN algorithm which achieves this behaviour [156]. As such EANN training within NEAT naturally generates increasingly complex classifiers, which can be extracted and used to create a cascaded structure.

For the combination of the outlined reasons the author has chosen to adopt an EANN approach for face detection. The training of the EANN is detailed in Chapter 4, whilst the software and hardware implementations for a mobile device is described in Chapter 5. The algorithm focuses on full frontal detection, with the simplistic assumption that multi-view face detection is a matter of using multiple detectors trained for different orientations [111][112]. This will be discussed further in Section 4.5. The final remaining decision is the choice of input features to use in conjunction with the classifier. Recently Haar-like features have become popular for desktop face detection processing [113][119][111][112]. However on a constrained computing platform, a number of issues make their choice less attractive. In particular, due to the potential for large accumulation values in the integral image, considerable memory storage is required. For example, Yang et al used 32-bits per element in the integral image. Viola uses 25 features in the first 2 layers of the cascade which removes 90% of the potential candidates. Nevertheless, the calculation of these 25 features uses more memory data bandwidth compared to retrieving the pixel intensity values directly. Therefore, pixel intensity values are used directly in this work. Although it is acknowledged that Haar-like features simplify the complexity of the classifier. Further investigation in this matter is targeted as future work.

3.3 Fundamentals of Evolutionary Artificial Neural Networks

Having decided to use EANNs as the basis of a face detection algorithm in this research, this section provides a context for the training and implementation details in Chapter 4 and Chapter 5 respectively. The fundamentals of ANNs are outlined in Section 3.3.1. This section coupled with the brief overview of GAs in Section 3.3.2 describe the fundamental elements of EANNs and as such this overview provides a context for the state of art review of EANNs in Section 3.3.3. The novel features of NEAT, the EANN open source library used in the proposed face detection algorithm are described in Section 3.3.4.

3.3.1 Artificial Neural Networks

ANNs are a versatile non-linear statistical data modelling technique, which take inspiration from the massively parallel biological network of neurons (consisting principally of axons, dendrites and synapses) in the human brain. The ANN consists of an interconnected network of simple neurons (sometimes also referred to as nodes). Each neuron typically has one or more input synaptic connections (also simply referred to as connections or links) and a single output. The connections originate from the outputs of other neurons and similar to the biological behaviour, the output value of the neuron is altered by a weighting in the connection before it reaches the input of the subsequent neuron. Neurons can be classified as input, hidden or output depending on their position within the network. Input neurons (sometimes known as sensors) receive values from the external world, but do not manipulate these values in any way. The value of output neuron(s) represent the result of the neural network processing and this value(s) are returned to the system in which the ANN is operating. Neurons whose inputs and outputs are not accessible from the outside system are termed hidden neurons and are typically necessary to solve non-trivial problems. Traditionally the neurons are organised into layers, (i.e. input layer containing input neurons, hidden layer(s) with hidden neurons and similarly the output layer contains output neurons), although as will be shown in Section 3.3.3 this is less applicable to EANNs. Basic terminology associated with ANNs is shown in Fig. 3.9.

The neurons can be organised into many different configurations or topologies. The simplest topology is a termed a single layer neural network, where there is only input neurons feeding directly to output neurons. This is however limited in terms of the types of systems it can approximate [157][158]. The notion of information flowing in a single direction from the input to the output is known as a feed forward network. Furthermore, if all the neurons in a layer are connected to all the neurons in the adjacent forward layer, the network is then said to be a fully connected multilayer feedback forward network. This is a popular topology for image classification problems including face detection [84][116]. It is also possible for feedback between neuron layers, this is said to be a recurrent ANN. An example of this is the Hopfield ANN [157][158].

In the classical model of a neuron[3], the output of the neuron j, is a function (known as the activation function and denoted as $f_{act}()$ in Eqn 3.1) of the summation of the weighted inputs to neuron j plus a bias value (see Eqn. 3.1) [157][158]. The bias can be considered a means of

[3]More advanced neuron models can have more complex relationships between the inputs and output. For example, the output of a spiking neuron also has a dependency on the timing of the inputs.

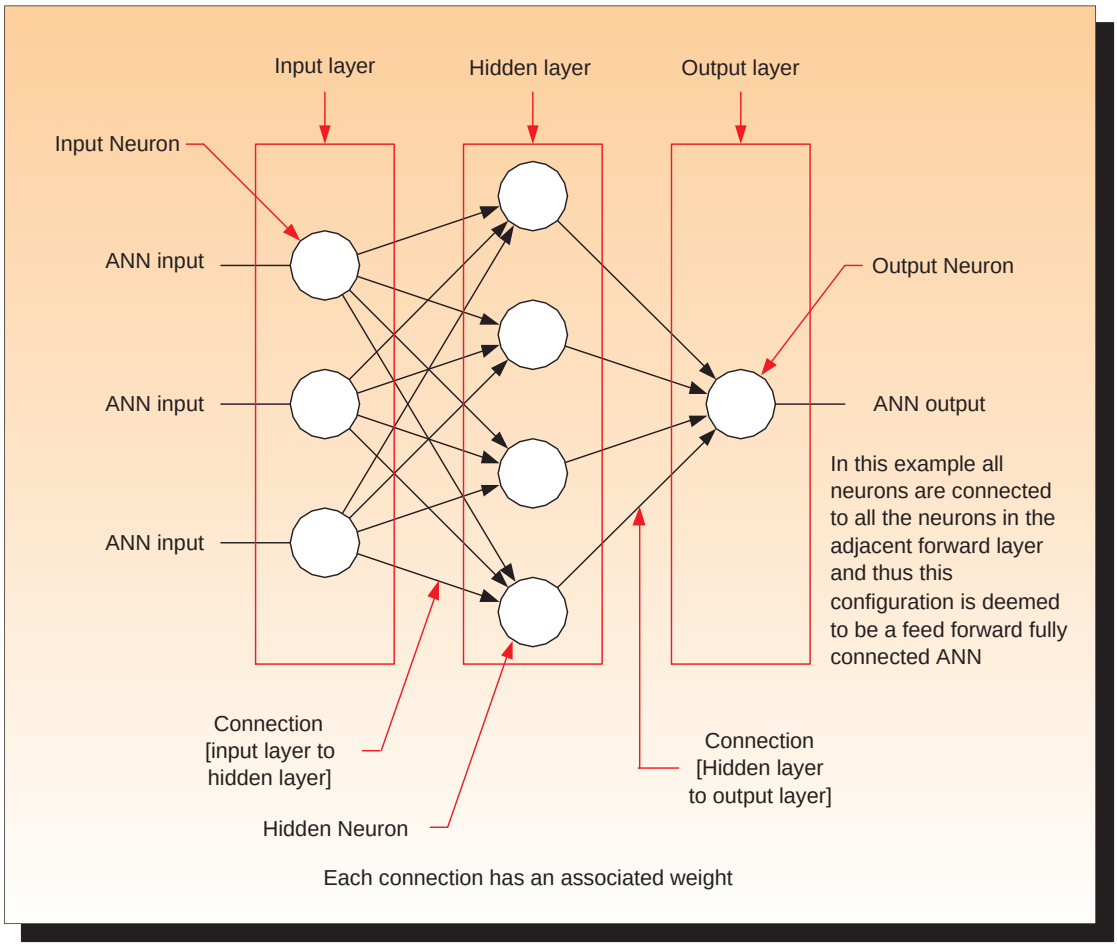


Figure 3.9: Basic Terminology associated with Artificial Neural Networks

changing the activation threshold. Commonly employed activation functions include threshold based (see Eqn. 3.2), ramp based (see Eqn. 3.3), sigmoid (also known as logistic function, see Eqn. 3.4) and Tanh (see Eqn. 3.5). These activation functions are shown in Fig. 3.10. Sigmoid functions are very commonly used and unlike the threshold function the sigmoid is differentiable, which makes it more amendable to a wider variety of training algorithms. The output of the neuron then propagates to the input of many other neurons and the process repeats until the final output layer of neurons is reached.

$$y_j = f_{act}\left(\sum w_{ij}x_{ij} + b_j\right) \tag{3.1}$$

$$y_j = \begin{cases} 1 & \text{if } y\_in_j \geq \text{T} \\ 0 & \text{if } y\_in_j < \text{T} \end{cases} \tag{3.2}$$

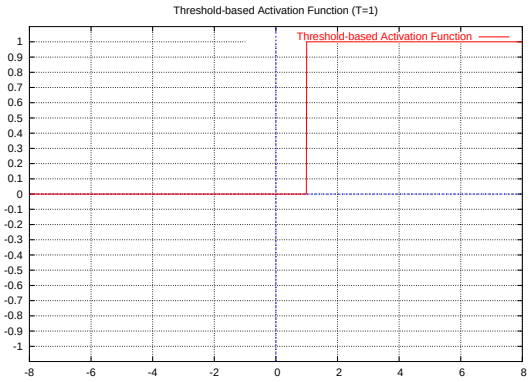
$$y_j = \begin{cases} 0 & y\_in_j < -\frac{z1}{2} \\ & -\frac{z1}{2} < y\_in_j < \frac{z1}{2} \\ 1 & y\_in_j \geq \frac{z1}{2} \end{cases} \tag{3.3}$$

$$f(x) = \frac{1}{1 + e^{-\sigma x}} \tag{3.4}$$

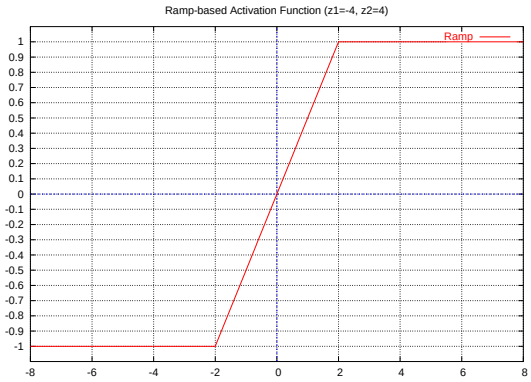
$$f(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} \tag{3.5}$$

Prior to use, an ANN must undergo a training phase. The training attempts to extract an underlying relationship from a set of observations (e.g. the input training samples and the output(s)). If this process is successful, the ANN is able to generalise the relationship from the training data to give a correct result for future data samples, even if these samples were not part of the initial training data. There are many different training algorithms, but these can all be broadly classified into the following families: supervised learning, reinforcement learning and unsupervised learning. Normally any topology can be used in conjunction with each type of training algorithm.

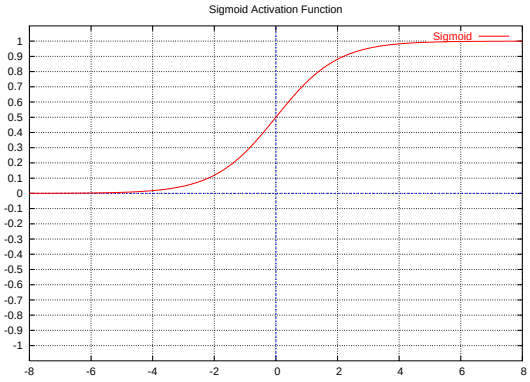
Supervised learning attempts to minimise a cost function for a given set of example pairs (i.e. inputs and the associated correct output value). Based upon the cost function (e.g. mean squared error between the generated output value of the ANN and the expected output value) the ANN is adjusted. Typically this adjustment is to the weight values, though other free parameters associated with the ANN could also be adjusted depending on the training algorithm. Supervised training has been shown to be applicable for many application domains including image pattern recognition/-



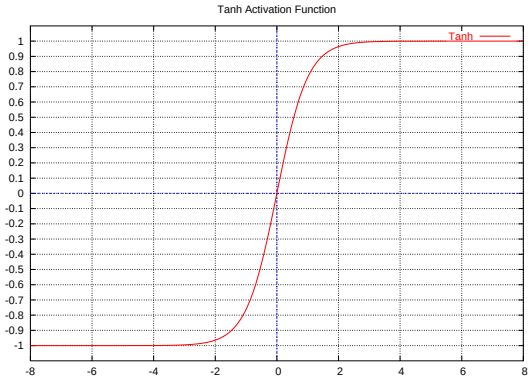
(a) Threshold-based activation function



(b) Ramp-based activation function



(c) Sigmoid activation function



(d) Tanh activation function

Figure 3.10: Selection of popular activation functions

classification and function approximation. Back-propagation is probably the most well known supervised training algorithm. This minimises the cost function using gradient descent and a generalisation of the least mean square rule. However, due to the gradient descent based approach, there is a potential for the training to get stuck at local minima. A further issue in general with supervised learning is that it is only suitable for problems where there is a precise target output value available. However, in many cases this information is not available. In such situations reinforcement learning can be used. This approach learns via trial and error interactions with a dynamic environment and the associated consequences of actions rather than explicit teaching (e.g. with marked up outputs). Reinforcement learning has been used successfully in control applications and artificial intelligence for computer gaming [31]. Finally, unsupervised learning attempts to automatically map input points to an output space without explicit correct output results available to guide the training process. It is frequently used in clustering-type applications, including image segmentation [153]. The popular Kohonen Self Organising Maps (SOM) algorithm uses unsupervised training. In general the most suitable learning paradigm becomes clear from the type of application. For example, in the case of the proposed face detection algorithm in Chapter 5, due to

the readily available large collection of sample training samples and the associated correct target values, a supervised training approach was the obvious learning paradigm to adopt.

3.3.2 Genetic Algorithms

A GA can be considered a type of search or optimising algorithm inspired by biological evolution. It is one member of the wider family of EAs, which encompass genetic algorithms, genetic programing, evolutionary programming and evolution strategies. The key feature of a GA is that the search space is explored in multiple locations using a population of genomes (frequently referred by other names including individuals, structures, etc) and these genomes are allowed to reproduce, which ideally creates offspring that converge toward the global solution. Each genome is an abstract representation of a candidate solution (typically termed a phenotype). Traditionally the genomes are represented using a binary encoding scheme but other formats such as integer and floating point are now also popular. Simple pseudo-code for a generic GA is shown in Algorithm 1, one iteration of the *while* loop is termed a generation.

Algorithm 1: Genetic Algorithm pseudo-code

```
initialise population;
evaluate fitness of population();
while termination_condition != true do
    select parents from population;
    crossover();
    mutation();
    evaluate fitness of population();
```

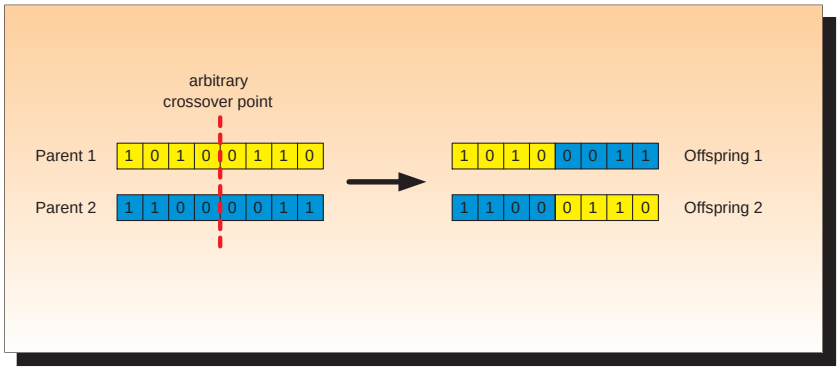
Each generation consists of a reproduction process, in which the GA uses the biologically inspired operators of selection, crossover and mutation (see Algorithm 1). Selection is used to choose parents for a new offspring. There are many different selection algorithms, including tournament selection, roulette selection, elitist selection, etc [159]. Roulette selection was used in the EANN for the proposed face detection algorithm in Chapter 5. This scheme assigns a higher probability to fitter parents reproducing but also deliberately allows unfit solutions through in order to preserve diversity in the population. After the selection process, crossover (sometimes known as recombination or mating) is the mechanism in which two parents[4] reproduce to create an offspring. There are many possible crossover schemes, principally involving swapping some elements of data between the parents. Algorithms include single-point crossover (see Fig. 3.11(a)),

[4]Some crossover mechanisms use more than two parents. For example simplex binary crossover scheme, uses two "fit" parents and one "unfit" parent. When the fit parents agree on the value of a bit the offspring takes this value. However when there is disagreement on a bit value, the offspring takes the opposite value of "unfit" parent.

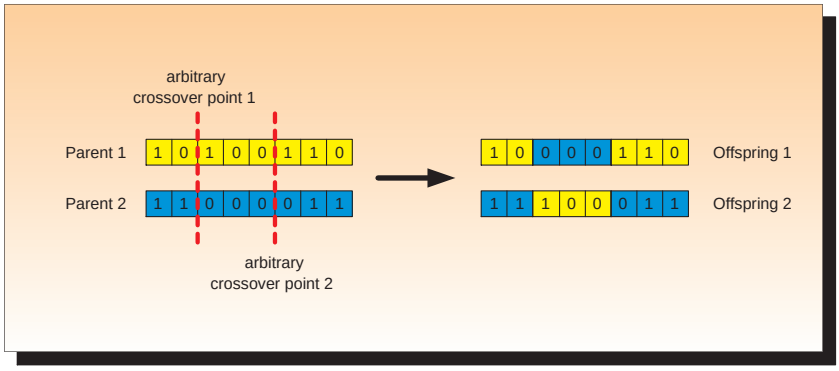
two-point crossover (see Fig. 3.11(b)), multipoint crossover 3.11(c), uniform crossover, etc [159]. After crossover (or instead of crossover depending on parameter probabilities) mutations are used to create random perturbations within the offspring. The classical example of mutation is bit flipping in a binary genetic sequence. Mutation maintains diversity in a population by preventing the population from becoming too similar to each other. The benefit of this is that it avoids local minima. However, if the frequency of mutation is set too high, the search effectively becomes a random search which can make it difficult to converge to a solution.

3.3.3 Evolutionary Artificial Neural Networks

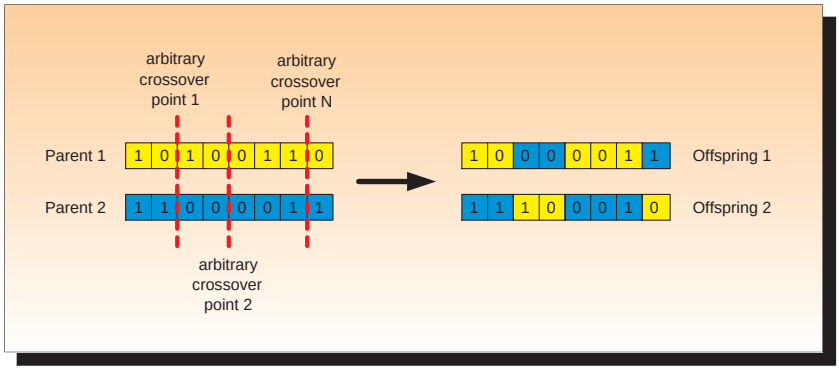
An alternative ANN training approach is to use an EA to evolve a solution within the ANN architecture search space. An ANN trained in this manner is usually referred to as an EANN. Typically a GA is used in the evolution but other EAs can be used, including genetic programming [160]. EANN algorithms differ in terms of the evolution target within the ANN. The simplest EANN approaches evolve only the weights in a fixed topology whereas the more advanced approaches evolve the architecture (i.e the connectivity, number of neurons, neuron activation function etc.) and provide the scope to adapt the learning rules [161][156]. The EANN training paradigm exploits the inherent ability of EAs to search large complex non-differentiable and multi-modal search spaces [161]. In the absence of such an automated approach for optimum topology selection, this extremely challenging task is traditionally approached in an ad-hoc manner of human trial and error. As well as being time consuming, such an ad-hoc approach has a high probability of selecting a non-optimum topology. There are other approaches for selecting topologies, including destructive training algorithms. These algorithms start with an over-sized ANN and progressively prune the structure backward. However, it has been shown that these approaches typically give inferior results, as only a very limited subspace within the potentially infinitely large search space is explored [162][163]. Aside from the considerable benefit of automatic architecture selection, EANNs have advantages over popular gradient descent based learning algorithms, such as back propagation and its many variants. Gradient descent based training has a tendency to get trapped at local minima, cannot be used for applications where the error function is non-differentiable and is unsuitable for sparse reinforcement type learning problems [161][156]. As such, EANNs could be said to be suitable for a wider range of application domains compared to ANNs trained via more traditional learning algorithms. This feature is particularly attractive from a dedicated hardware architecture perspective, as it maximises the reusability of such a hardware core.



(a) Single Point crossover



(b) Two point crossover



(c) Multi point crossover

Figure 3.11: Examples of genetic algorithm binary crossover techniques

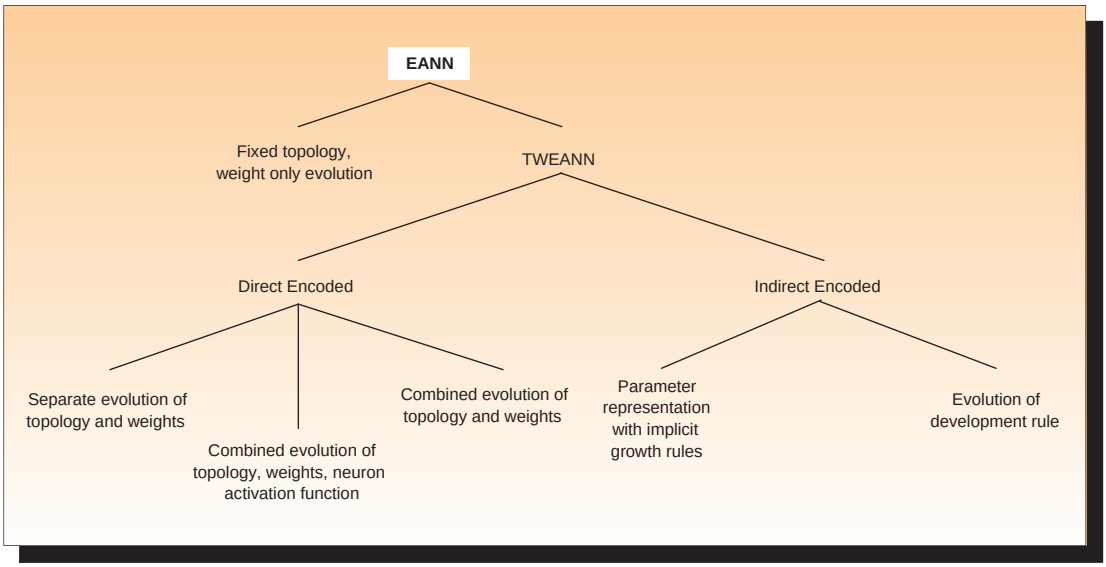


Figure 3.12: Taxonomy of EANN algorithms

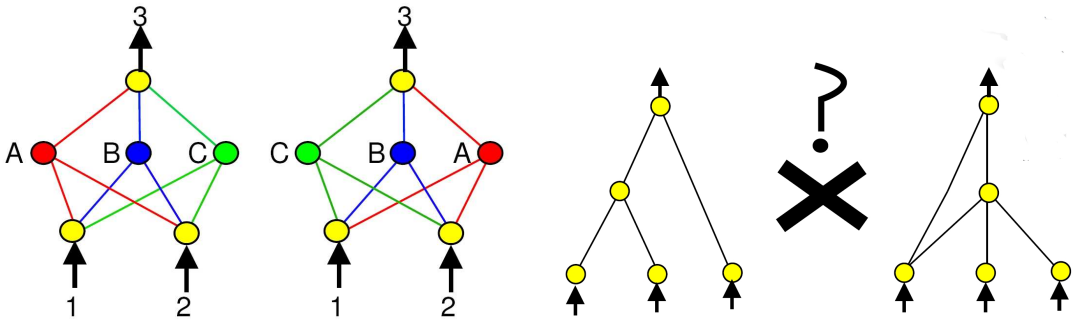
A taxonomy of the varied approaches for EANNs is shown in Fig. 3.12. The most straightforward EANN algorithms use a fixed topology and only evolve the value of the weights. In this case the genomic representation is a relatively straightforward decision between a binary representation or using real numbers. Using a binary representation is easier to implement, as it allows classical crossover and mutation operators. However, careful evaluation and trade offs are required to establish the number of binary bits to use to achieve adequate weight precision, without having an excessively long genome representation which effects the evolutionary speed. The alternative is to use real numbers, which minimises this issue, though the crossover and mutation operators become more complex.

The genome representation is considerably more difficult if the topology as well as weights are allowed to evolve. The two principal genome encoding schemes for this approach are direct encoding and indirect encoding. These could be considered to differ in terms of the amount of the structural information that is captured in the genome. Direct encoding schemes contain the information about every connection and every node in the genome. As a result, the genome representation can be quite large, although there is a one to one mapping between the genome and phenotype. Within direct encoding, a number of broad approaches have been suggested including genitor, binary matrix, node, etc. [164]. Node based schemes have a number of desirable characteristics including ease of scalability and are preferable for these reasons. The subsequent evolution of the population of direct encoded genomes can proceed in one of two ways. The first and easier of the two methods is to use incremental evolution of the topology followed at each

stage by weight training (EA-based or otherwise). The alternative is to simultaneously evolve both the weights and topology. The incremental approach is considerably slower and the separate training can introduce effects such as randomness resulting in a "noisy fitness", which can have a negative impact on the evolutionary process [160]. For these reasons, the simultaneous evolution of weights and topology is considered a better approach. The alternative to direct encoding schemes is indirect encoding. In these schemes only the most important details are stored in the genome representation. This produces a more compact genome. There are two main options for indirect genome representation. One option is to encode the main parameters, such as number of nodes in a hidden layer, etc. following which these parameters are then evolved. The detailed connectivity in this case is produced via explicit growth rules. The other option uses fewer parameters and directly evolves the growth/development rules. However, question marks remain over the ability of indirect encoding schemes to evolve a solution which gives good generalisation [161].

Regardless of whether direct or indirect encoding schemes are chosen, one of the most difficult issues to overcome when evolving a topology is the problem of competing conventions (also known as the permutation problem). The competing conventions problem stems from the fact that there may be more than one way to express a topology. An example situation is shown in figure 3.13(a), where although the position of neuron A and C are swapped, the networks are identical. This causes considerable trouble for the crossover operator, since identical parents are being crossed over, there is a high risk that functionality within the topology will be destroyed [156]. The effects can be so detrimental that some researchers choose not to use the topology crossover operator and instead rely on mutation. However, for many applications crossover has been shown to be vital [161]. Even without the complication of competing conventions, topology crossover is a difficult problem. How two topologies should be recombined is frequently not obvious (see figure 3.13(b)). There is also the possibility that crossover could give rise to an offspring with an invalid structure, for example a neuron with no input or output connections. Topology analysis can be used to minimise this effect.

The problem of competing conventions is less of an issue for indirect encoded genome schemes with evolutionary growth rules, though the next section outlines a recent direct encoded node based scheme which solves the problem. This scheme also allows meaningful topology crossover to occur without the computational overhead of topology analysis.



(a) The competing conventions problem (Source: Stanley [156])

(b) Difficulty of the topology cross-over operator (Source: Stanley [156])

Figure 3.13: Issues associated with topology genome representation & the crossover operator

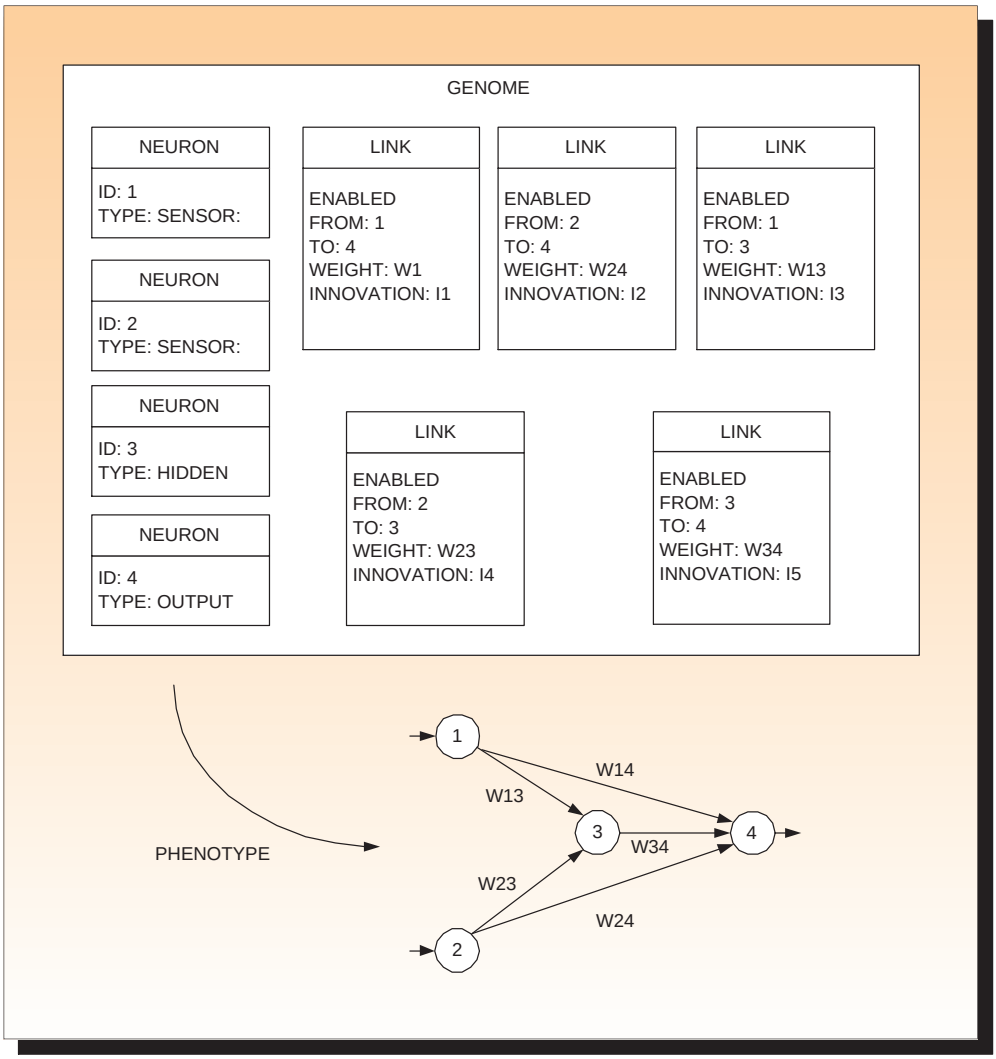


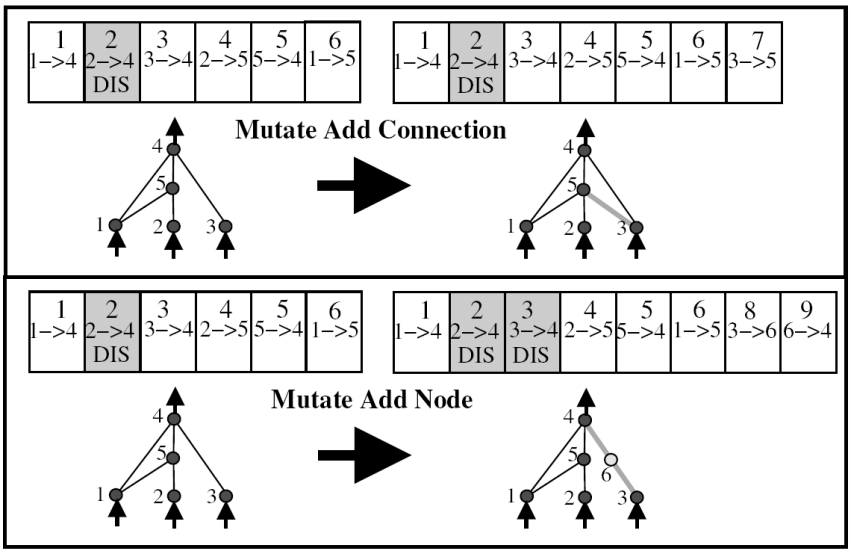
Figure 3.14: Example of a NEAT genome and resultant phenotype

3.3.4 Neuro Evolution of Augmenting Topologies

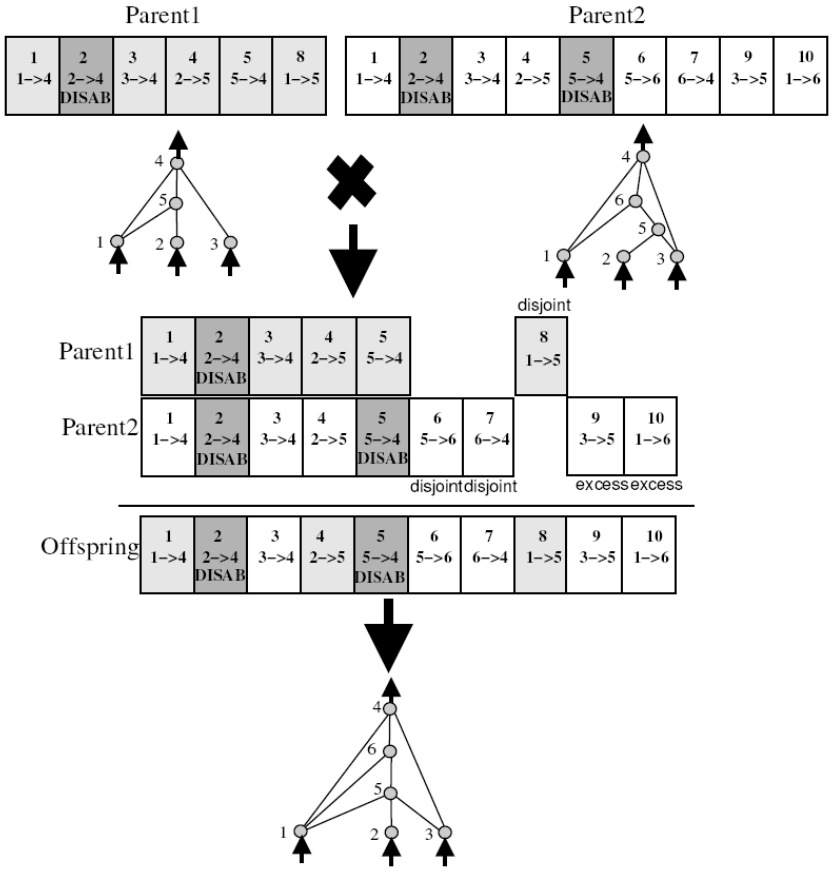
NEAT is an example of a direct encoded node based EANN approach, which is deployed in a broad spectrum of varied applications [156]. Each genome consists of a number of link and neuron genes as shown in Fig. 3.14. It uses a genome historical marking scheme, referred to as innovation numbers, which is the enabling factor for much of the novelty within NEAT and avoids many of the problems associated with other EANN approaches. Stanley found that recording the evolutionary history of each gene identifies the common structure between two genomes [156].

Tracking the evolutionary history is simply a matter of incrementing a global innovation number whenever structural mutation (addition of a connection or neuron) occurs. A unique innovation number is then assigned to the new gene. This simple scheme allows meaningful topology crossover to occur and does so whilst avoiding any computational overhead of topology analysis. An example of using innovation numbers in the structural mutation operators is shown in Fig. 3.15(a). The top number in each box corresponds to the innovation number and the bottom numbers shows the source neuron and the destination neuron for each connection. An *add connection* mutation between neurons three and five, is simply recorded and the innovation number increments to seven. If this mutation was followed by an *add node* mutation between neurons three and four on the original topology, the connection between neurons three and four (innovation number 3) is firstly disabled. Two new genes are subsequently added, representing the new connections and neuron. This structural mutation is recorded via the addition of innovation numbers eight and nine. If during mutation the same structure arises as previously encountered, the innovation numbers are not incremented. In this way all similar topologies are uniquely identified. This avoids the issue of competing conventions during crossover. The NEAT crossover operator is shown in Fig. 3.15(b). The two parents are aligned according to their innovation numbers. Common genes are simply inherited in the offspring. There are two methods to crossover the weights in the parents to generate the connection weight in the offspring. The connection weight is either selected at random from the two parents or the average weight from the connections in the two parents is used. Genes that are present in one parent but not in the other are labelled as disjoint and excess genes. As will be described shortly, the number of disjoint and excess genes are used when establishing the structural similarity between genomes.

Adding structure to the topology will typically initially reduce the fitness function, as smaller structures tend to optimise faster. Consequently, there is little opportunity for the new topology to survive [156]. Taking inspiration from biological evolution, where an innovation needs time to



(a) Mutation in NEAT. (Source: Stanley [156])



(b) Crossover in NEAT. (Source: Stanley [156])

Figure 3.15: Use of innovation numbers in the mutation and crossover operators in NEAT

reach its potential, NEAT uses a process of specification (also known as niching within the context of GA). This allows genomes to compete only against their own species and gives the new offspring a better chance of survival. The similarity between two genomes is given by Eqn. 3.6, where E is the number of excess genes, D is the number of disjoint genes, $\overline{W}$ is the average weight difference between genes and N is the number of genes in the larger genome. The coefficients c_1,c_2 & c_3 weight the importance of each element (the standard values in the NEAT library have been used unchanged in this research). After crossover and/or mutation, each offspring is compared against a representative member (chosen at random) from the parents species in the previous generation using Eqn. 3.6. If the result is less than a predetermined threshold (to be considered similar), the new offspring is placed into the parents' species, otherwise a new species is created.

$$\delta = \frac{c_1 E}{N} + \frac{c_2 D}{N} + c_3 \overline{W} \tag{3.6}$$

Another important concept in NEAT, which also takes inspiration from biological evolution, is the introduction of the concept of complexification. Using this scheme processing starts with a minimal topology with no hidden nodes and each input is connected to an output node. Successive generations systemically elaborate the complexity by adding neurons and/or connections. This is attractive in the context of mobile devices since minimal topologies are favoured, thus reducing computational complexity and power consumption for a given problem. During complexification topology innovation occurs. The combination of the features outlined allows NEAT to out-perform other EANN approaches [133], and for this reason it was chosen as the EANN library for the proposed face detection algorithm.

3.4 Conclusions

This chapter firstly presented a comprehensive review of face detection algorithms. A taxonomy of face detection algorithms was presented and particular emphasis was placed upon appearance-based algorithms and current hardware implementations. It was concluded from the review that appearance based algorithms were inherently more suitable for implementation in hardware compared to feature extraction based methods. The potential benefits of using an EANN based approach for face detection were outlined. Possibly the greatest benefit offered by an EANN for a mobile device face detection implementation, is the scope for using a minimal topology, which will reduce the overall power consumption. Having made the design decision to adopt an EANN

based approach, a thorough explanation of the underlying theory behind EANNs was presented. This provides context for the algorithmic implementation details in subsequent chapters.

CHAPTER 4

A Novel Face Detection Training Algorithm

This chapter outlines the training phase of the proposed Evolutionary Artificial Neural Network (EANN) based face detection algorithm. The goal of the training routine is to find an optimum network topology and weight values which will give acceptable face detection performance, whilst also striving for a minimal topology so as to reduce the ultimate computational cost. The training process occurs once, after which the final evolved solution can be deployed for face detection on the chosen platform. Since the training phase is not required during normal operation, the process can leverage all available computational resources. As such, this eliminates the hardware restrictions of the mobile device during the training phase. During this research a two dual-core AMD processor based server and a Pentium-4 based laptop were used. This chapter firstly describes the training data preparation. This is followed by an overview of the training routine and explanations for the various design decisions taken. A detailed account is then given of the training run variants and the rationale used for parameter selections. The chapter concludes with a review of the training results, as well as observations and general recommendations for training an EANN for a face detection application.

4.1 Training data preparation

Four freely available and commonly used face databases (FERET [129], CMU [84], Yale [115] & BioID [165][166]) were used in the generation of the proposed face detection training algorithm.

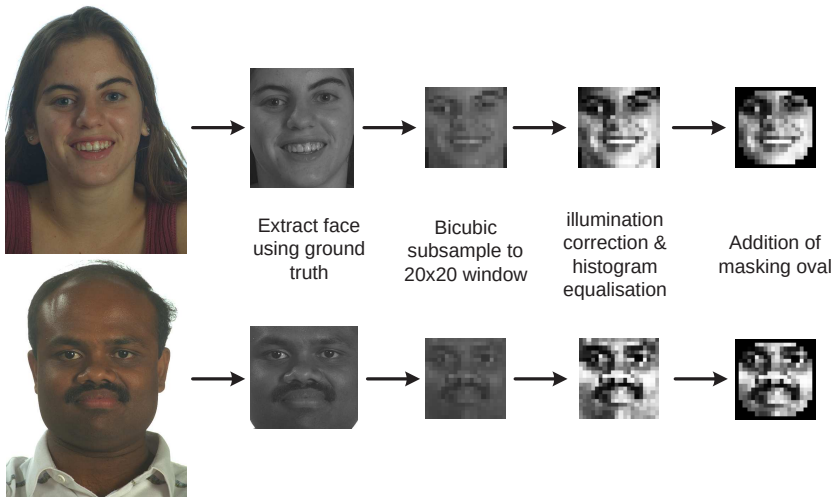


Figure 4.1: Preprocessing of FERET face training dataset

These face databases were separated into 3 datasets, a training dataset, a valuation dataset and the actual testing dataset. The training dataset consists of 942 greyscale faces images in a frontal view taken from the FERET face database [129]. Each face has a marked up location for the eyes, nose and mouth. These coordinates are used to generate a new 20×20 pixel image, which is a cropped and subsampled (bi-cubic) version of the original image. Prior to being used as a positive training sample, the new image undergoes illumination correction, histogram equalisation and has an oval mask fitted, in a manner similar to that proposed by Sung et al [115]. The purpose of the oval mask is to eliminate any unnecessary background structure. Additionally, it also helps in reducing the dimensionality of the Artificial Neural Network (ANN) due to fewer inputs. These processing steps can be seen in Fig. 4.1. A general guide when using ANNs is that the total number of positive and negative training samples should be 5 to 30 times the number of connections [167]. A typical number of connections for a face detection ANN is 3,000 – 4,000 [84]. Therefore, to increase the positive training set size, rotated (± 5 degrees with bi-cubic interpolation) and mirrored versions are also generated for each sample.

Generating representative non-face training data is generally accepted to be more troublesome [115]. As a consequence, a bootstrapping scheme similar to that used by many authors is employed in this work [115][84][116][83]. This leverages randomised data for initial non-face samples. Then after a period of training, the best solution at that point is evaluated with a dataset that has no faces present (e.g. a scenery image such as that shown in Fig. 4.2). The resultant incorrect false positives (i.e. non-faces classified as faces) are added to the training dataset as examples of non-face data samples. This is used to increase the size of the non-face dataset, which in turn improves

Figure 4.2: Example scenery image devoid of faces which was used during the training image

the precision of the face detection results. These steps are outlined in Algorithm 2. It should also be noted that the bootstrapping process can also be used to add faces to potentially improve the recall rate (i.e. the percentage of correct faces detected). The frequency of bootstrapping and the number of non-face samples added per iteration are described in further detail in Section 4.3. As with the positive face samples, each non-face sample undergoes illumination correction, histogram equalisation and has an oval mask added (for consistency purposes).

Algorithm 2: Non-face bootstrapping pseudo-code

```
open image file(s) devoid of faces;
while (bootstrap_cnt < bootstrap_max && (!(all 20x20 pixel blocks in images examined))
do
    extract 20x20 pixel window;
    illumination_correction(window);
    histogram_equalisation(window);
    add_oval_mask(window);
    classification_result=0;
    for (i=0; i<=ensemble_max; i++) do
        network(i)→load_inputs(window);
        network(i)→activate();
        classification_result+=network(i)→output;
    if (classification_result/ensemble_max)>=faceDet_threshold then
        save 20x20 window as "bs_tdata_" + bootstrap_count + ".pgm";
        add "bs_tdata_" + bootstrap_count + ".pgm" to non-face-training-data.txt;
        bootstrap_count++;
```

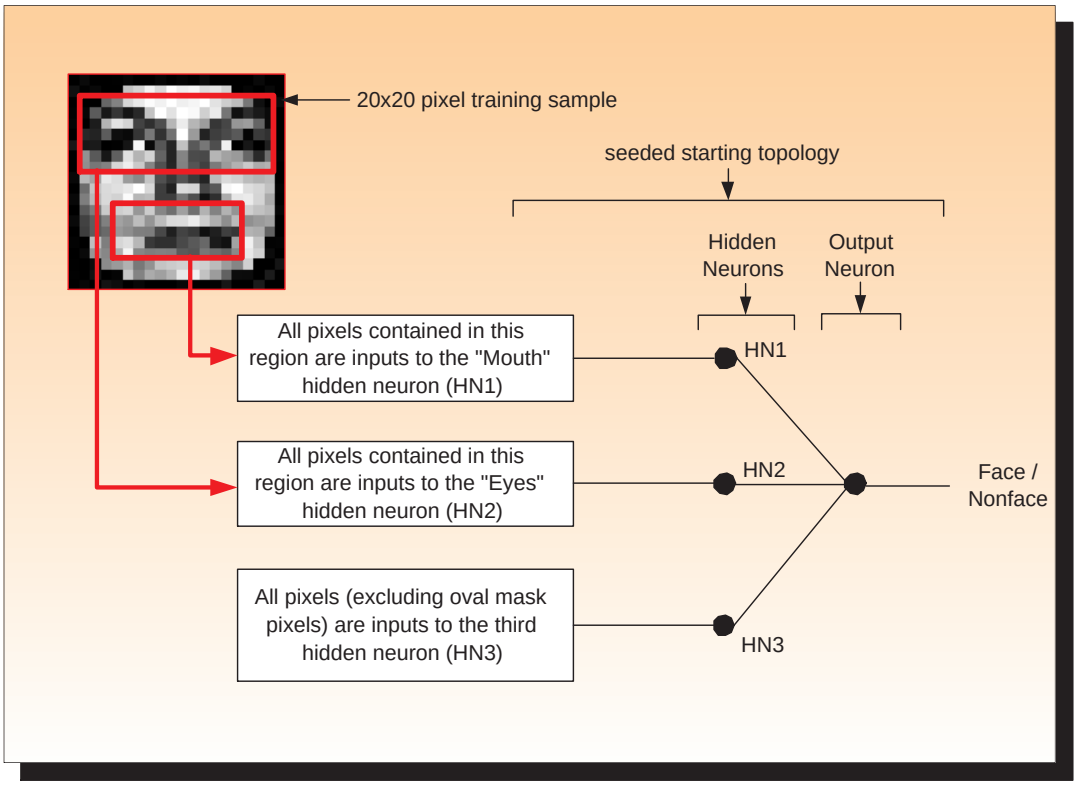


Figure 4.3: Starting topology seeded with localised features

4.2 NEAT based face detection training

During the initialisation of the algorithm, the Genetic Algorithm (GA) (which controls the evolutionary process), spawns an initial population of N_{pop} genomes, from an input topology file containing a single genome. N_{pop} is chosen to be as large as possible within the constraint of acceptable run times, therefore N_{pop} values ranging from 50 to 200 were explored during training. The input topology file typically represents the minimal topology for the given problem, usually meaning that all input neurons are directly connected to one or more output neurons without the presence of hidden neurons. Neuro Evolution of Augmenting Topologies (NEAT)-based complexification progressively evolves the solution from this minimal starting point. For experimentation purposes, an additional seeded topology with hidden neurons present was also used as an initial starting topology in a separate training stream. Details of the two separate seeded and non-seeded training streams are presented in Section 4.3. The seeded starting topology used had three hidden neurons and one output neuron. One hidden neuron was fully connected to all inputs, whilst the remaining two hidden neurons used input pixels from regions around the eyes and mouth respectively (see Fig. 4.3). This configuration is chosen to investigate if seeding the starting topology with localised features leads to the evolutionary process developing a more robust solution.

Once the initial population is spawned, face detection training starts by randomly selecting X_{face} and $Y_{nonface}$ training samples from the training dataset. The value of X_{face} and $Y_{nonface}$ is a trade-off. With infinite computational resources, all samples within the face and non-face training datasets could be applied each generation. However, it was found that even when using moderate size populations (100) on a high specification server (equipped with two dual-core 2.2-GHz AMD 64-bit processors and 10-Gb of RAM) this approach leads to excessive run times (in the order of a week for 500 generations). Consequently, a randomised subset of the training dataset is applied each generation. This approach is intuitively reasonable as there should be a high degree of correlation between all samples in the face training dataset. The application of a subset of training samples is also explicitly mentioned in the work of Garcia et al [83]. The randomised nature of the selection avoids introducing any bias from applying the training data samples in a fixed sequence. Furthermore, whilst the randomised selection of faces and non-faces is fixed for a generation, the order of these training samples is randomised each time they are applied to a genome to further ensure that no bias is introduced. Values ranging from 5 to 5000 were explored for X_{face} and $Y_{nonface}$. Smaller values provided less granularity of fitness between genomes and as such the evolution had a tendency to get stuck in local minima, whereas large values impacted the run time. It was found that values in the range of 50 to 300 offered an adequate trade-off. This range is similar to the value of 50 used in the convolution neural network based face detection work of Garcia [83]. The exact selection of X_{face} and $Y_{nonface}$ used during each training run is discussed in Section 4.3.

The subset of training samples are then iteratively applied to each genome in the population. For each sample, only the non oval mask pixels are used. This reduces the number of inputs from 400 (i.e. 20×20 training sample) to 292. When each sample is applied, the genome is activated and the value of the output neuron is calculated. Using this value and knowledge of whether the training sample was a face or non-face an error is calculated, squared and accumulated. This process repeats until all of the samples are applied to this genome. Then using the accumulated error, a mean squared error based fitness value is calculated for the genome. This process repeats for all genomes in the population. These processing steps can be seen within the two ***while-loops*** in the pseudo-code in Algorithm 3. It should be noted that the fitness function shown in Algorithm 3 makes use of an optional weight decay penalty term. This penalises the fitness of networks with larger absolute valued weights, thus favouring networks with smaller absolute valued weights. It has been claimed that the size of weights is more important than the number of weights, and

that larger weights can actually impinge the generalisation ability of a network [168]. This has not been previously explored in the context of the Neuro Evolution of Augmenting Topologies (NEAT) library, and so was deemed a useful experiment. As can be seen in Algorithm 3 the different types of connections (input to output, input to hidden, hidden to output & hidden to hidden) in the network require different weight decay parameters. The choice of these parameters, as well as conclusions about the overall usefulness of combining weight decay with the NEAT library is discussed in Section 4.4.

The GA in the NEAT library then evolves the population using mating and/or mutation. It should be noted that the functionality of the GA has not been altered from the standard NEAT library and as such there is no contribution in this area. However, for completeness a brief overview of how the GA functions in NEAT is now described. Using the fitness of each species in the population, NEAT decides on an appropriate number of offspring in each species for the new population. A check is then carried to ensure population level stagnation is not occurring. If detected, delta coding is performed, this reinitialises the population but attempts to preserve the evolutionary progress to date [169]. After this, based principally upon input GA parameter settings (*MUTATE_ONLY_PROB* & *MATE_ONLY_PROB*), mating and/or mutation is performed. During mating, the choice of parent genomes within a species is decided using roulette selection. Randomised selection within the fittest species was also explored and gave comparable results due to the relatively small species size, this observation was also noted by Stanley [156]. Single point crossover, multi-point crossover and multi-point average crossover are the available crossover mechanisms, again input GA parameters (*MATE_MULTIPOINT_PROB*, *MATE_MULTIPOINT_AVG_PROB* & *MATE_SINGLEPOINT_PROB*) decide upon which mechanism to use at any instance in the evolution. Interspecies mating is also allowed, but typically the probability (*INTERSPECIES_MATE_RATE*) is set very low. After mating (or instead of mating based upon the *MUTATE_ONLY_PROB* parameter) the offspring can be mutated. The supported mutation operators and associated input probability parameters are: add neuron (*MUTATE_ADD_NODE_PROB*), add connection (*MUTATE_ADD_LINK_PROB*), disable connection (*MUTATE_TOGGLE_ENABLE_PROB*), re-enable connection (*MUTATE_GENE_REENABLE_PROB*) and perturb weights (*MUTATE_LINK_WEIGHTS_PROB*). After mutation, the offspring is placed into the correct species or a new species is created if none are within the input parameter threshold (*COMPAT_THRESH*) bounds. The mating and/or mutation continues until the new population is complete.

Algorithm 3: Simplified EANN-based training algorithm for face detection

```
pre_process_all_training_samples;
initialise X_face;
initialise Y_nonface;
initialise weight decay constants α, beta, γ, δ;
if continue_training then
    load_population(partially_trained_population_input_file);
else
    spawn_initial_population(starter_genome_file,N_pop);
while (stopping_condition() != true) do
    randomly select X_faces faces from full face training dataset;
    randomly select Y_nonfaces non-faces from full non-face training dataset;
    for all genomes in population do
        error_accum = 0;
        while not all X_face face and Y_nonface non-face samples are applied do
            randomly select face or non-face training sample;
            apply training_sample to inputs of genome network;
            for (i=0; i<=network_depth; i++) do
                genome→network→activate();
            if ground_truth(training_sample)==“face” then
                if genome→network→output>=face_threshold then
                    true_positive++;
                    error=1-genome→network→output;
                else
                    false_negative++;
                    error=(1-genome→network→output)× false_negative_penalty;
            else
                if genome→network→output<face_threshold then
                    true_negative++;
                    error=genome→network→output;
                else
                    false_positive++;
                    error=genome→network→output × false_positive_penalty ;
            error_accum += squared(error);
        MSE = error_accum / (X_face + Y_nonface);
        genome→network→find_different_weight_totals(in2out_weight_sq,
        in2hid_weight_sq, hid2out_weight_sq, hid2hid_weight_sq);
        weight_decay_penalty =
        α×in2out_weight_sq+β×in2hid_weight_sq+γ×hid2out_weight_sq+δ×hid2hid_weight_sq;
        genome→fitness = 1 / (1+MSE+weight_decay_penalty);
    population→ga_evolve();
```

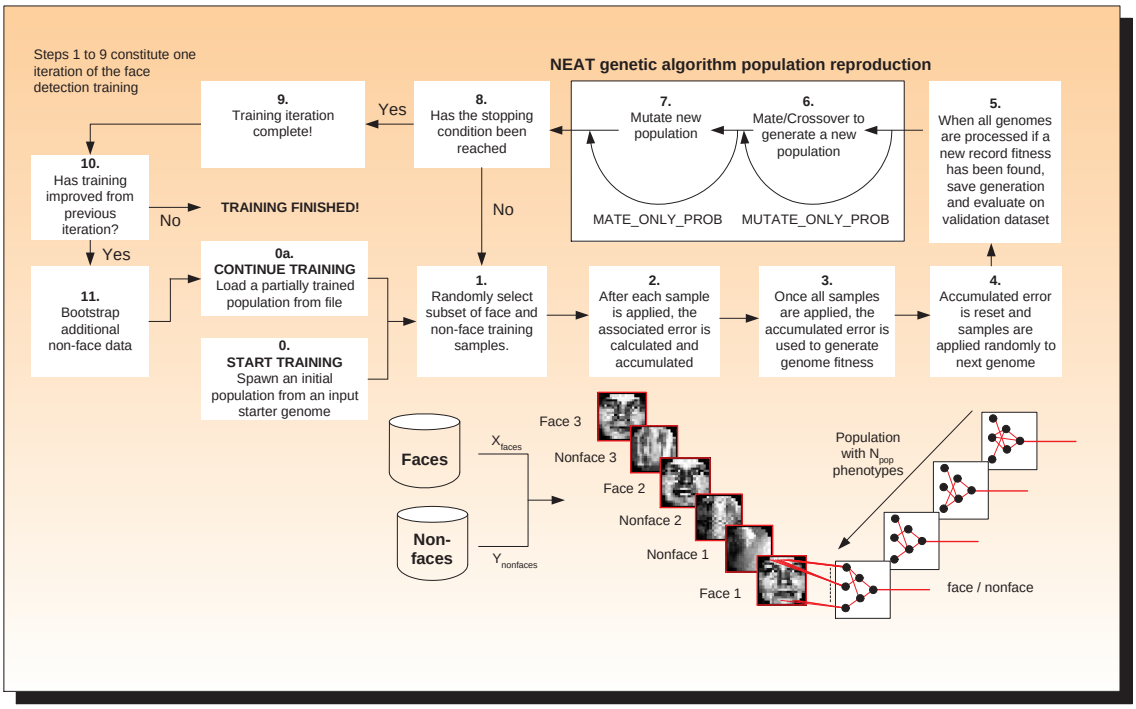


Figure 4.4: Proposed NEAT-based face detection training

The full training algorithm is illustrated in Fig. 4.4. This diagram shows the NEAT functionality relative to the overall proposed face detection training. Whilst the GA within NEAT was not modified, considerable influence can be exerted on the evolutionary process through the choice of the GA parameter selection. Furthermore, using techniques such as changing the relative values for X_{face} and $Y_{nonface}$ and using uneven penalisation of incorrect false positive and false negative classification results can greatly influence how the search space is explored. For example, by applying many more non-faces compared to faces, initial phases of the evolutionary process typically label all samples as non-faces before gradually classifying faces correctly. Additionally, if a false positive is penalised heavier than a false negative, the areas of the search space yielding higher precision are explored more thoroughly, albeit at a potential cost of reduced recall. The selection of values for the GA parameters, X_{face}, $Y_{nonface}$ and incorrect classifications penalty terms, which were used during each training run is discussed in detail in Section 4.3.

The process of population face detection fitness evaluation and population reproduction keeps repeating until a stopping condition is satisfied (step number 8 in Fig. 4.4). The simplest stopping condition is to enforce an upper limit on the number of generations in which evolution is possible. However, deciding this in advance is difficult and indeed somewhat futile due to the inherent randomness within the GA search, which naturally leads to varying training duration if run multiple times. Not training for long enough leads to under-fitting of the training data and con-

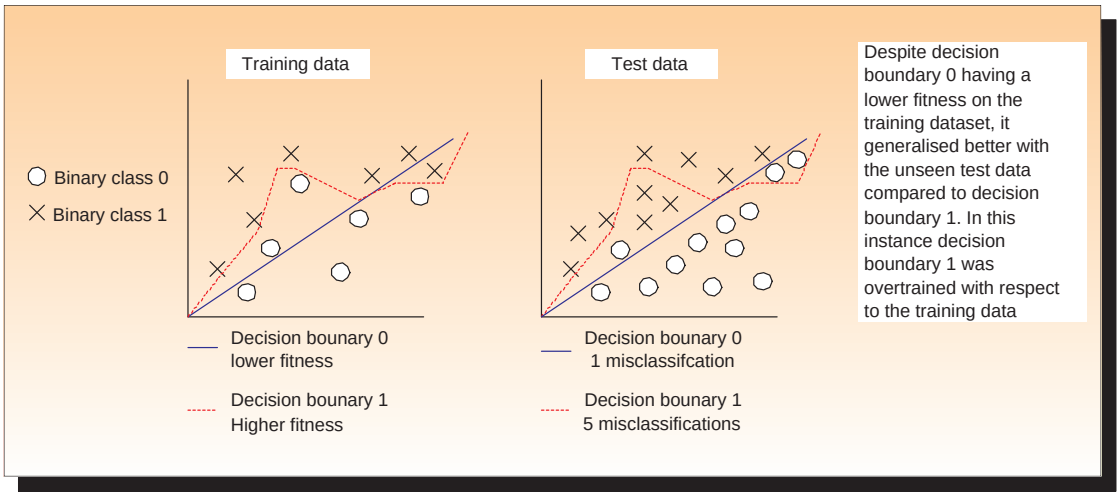


Figure 4.5: Simple example of over-training

sequently poor generalisation performance with unseen data, whilst training for too long leads to the over-training phenomena [158][157][167]. Another obvious (although problematic) approach is to keep training until an arbitrarily small error is recorded in the training dataset. The problem with this approach is that beyond a point, the training process risks over-fitting the training dataset and any noise present in the data. The consequence of this is a loss of generalisation ability with unseen data. A simple demonstration of overtraining is shown in Fig. 4.5. Clearly, more advanced techniques are needed for the face detection training stopping condition.

For good generalisation performance (avoiding under and over fitting of training data) it is generally recommended to at least remove statistical outliers from the training samples and use a large enough training dataset (both techniques have been employed in this work) [167]. Furthermore, there are a variety of common methods employed for achieving good generalisation performance. These include model selection, jittering, early stopping, weight decay, Bayesian learning, combining networks, etc. [167]. Weight decay and the combination of networks have been investigated in this work and are discussed in Section 4.3 and Chapter 5 respectively. In addition, early stopping was investigated to see if it was a suitable candidate for the stopping condition for the face detection training. The early stopping technique periodically evaluates the best interim solution on a separate dataset called a validation dataset. When the error associated with the validation dataset starts to increase relative to the previous evaluation of the validation set, the GA stopping criteria is reached and training stops regardless of whether the training error is still reducing in the training dataset. The increasing error in the validation dataset indicates the training process is beginning to over fit the training data. This explanation of over-training assumes the error function is mono-

tonically decreasing. Unfortunately, as will be clear from the training recall and precision graphs in Section 4.3, the EANN face detection training error does not exhibit monotonically decreasing behaviour.

Due to this characteristic, the author explored using a smoothed version of the validation error as an indicator of over-training. Whilst this improved performance, it was still prone to terminate the evolutionary process too early. There are many reasons for this, the most obvious being that it is difficult to predict the duration of the smoothing "window". For example, if the evolutionary process explored local minima in the subset of the training dataset which were not present in the validation dataset, this could lead to an extended period of increasing fitness in the training dataset but a reduction of the fitness in the validation dataset. This scenario would trigger termination of the training, if the smoothing window duration was not long enough to capture the effect of a possible mutation in a later generation moving the training away from the local minimum and toward the global minimum. To give a clearer demonstration of this, consider the following scenario (albeit slightly contrived). If a high percentage of the generational non-face training subset was particularly difficult (almost face like), the fittest genome solution would most likely sacrifice recall for precision, particularly if $Y_{nonface}$ was greater than X_{face} and/or false positives were more heavily penalised. If this leads to a new record fitness, the drop in the recall ability of this genome could cause a reduction in fitness in the validation dataset. If subsequent record fitnesses are found around this local minimum in the training dataset this could cause a condition where there is a sustained drop in fitness in the validation dataset. This scenario could then trigger the early stopping.

As a consequence of the outlined issues, early stopping (i.e. in the sense of stopping when the validation dataset error increases) was not used in the final face detection training. Instead, the stopping condition favoured in this work used a combination of a maximum generation count and a target generalisation error. This generalisation error was estimated by evaluating the fittest genome on the validation dataset whenever a genome with a new record fitness was found in the training dataset. There are numerous approaches to generating a validation dataset, the method used is commonly referred to as split-sample or hold out validation. This reserves part of the training set to estimate the generalisation error. In order to get a good estimate of the generalisation error it is vital that the reserved subset of the training set is not used in any way during the training. One disadvantage of split-sample validation is that it reduces the size of the training set. However, due to the availability of multiple face databases, the author considered this was less of issue for the

chosen application. Had it been a concern, other approaches such as k-fold cross-validation could be considered [167]. The face samples used to generate the split-sample validation dataset were taken from the FERET database (samples different from those used in the training dataset) [129]. The non-face validation samples were taken from a scenery image without faces present.

During the training process, if a new winning genome is found, the entire population from that generation is stored so that it could be used later for regular mode face detection or as a partially-trained starting point to recommence training. When the stopping condition is reached, it is then assumed that the EANN will have found the best topology during the evolutionary search process for that given training dataset and set of configuration parameters. To improve the detection performance, it is possible to use further iterations of the training. For example, the next iteration uses the data samples from the initial training run and augments those with targeted training samples created from misclassification errors generated using the current best evolved solution. In this way the detection performance is iteratively improved by learning from its mistakes. This process is called bootstrapping and was described in Section 4.1 and is shown as step 11 in Fig. 4.4. Previous face detection ANN-based research, have iteratively trained for a predetermined number of epochs and then increased the number of non-face data samples via a bootstrapping technique [84][83][115]. Whilst this approach was investigated, the author found that within the EANN framework in general better performance was observed when the bootstrapping occurred after the stopping condition was reached. This is most likely caused by the inherent variability of the GA, which makes it difficult to predict in advance the number of generations required for training to reach an adequate performance level for a given dataset. In the proposed approach, after bootstrapping, a *continue_training* function facilitates additional training with the extra bootstrapped non-face training samples by allowing training to recommence from any previous generation (see step 0a in Fig. 4.4 and also Algorithm 3). The frequency of bootstrapping and the amount of non-face samples added per bootstrapping iteration are features of the distinct training strategies explored and is detailed next in Section 4.3.

4.3 Training Runs and Parameter Selection

It should be clear from the description in Section 4.2, that there are a large number of parameters that can be adjusted. These are a combination of NEAT GA parameter settings and general algorithm parameters (see Table 4.1 for a complete listing). General rules of thumb can assist in

Table 4.1: Tunable parameters in proposed face detection training scheme

Code Location	Parameter	Description
NEAT GA	weigh_mut_power	Weight mutation power
	recur_prob	probability of recurrent connections
	compat_thresh	Species compatibility threshold
	mutate_only_prob	mutate only probability
	mutate_link_weights_prob	Probability of mutating weights
	mutate_toggle_enable_prob	Probability of disabling a connection
	mutate_gene_reenable_prob	Probability of re-enabling a disabled connection
	mutate_add_node_prob	Add new neuron mutation probability
	mutate_add_link_prob	Add new link mutation probability
	interspecies_mate_rate	Probability of mating between species
	mate_multipoint_prob	Multipoint mating probability
	mate_multipoint_avg_prob	Multi-point average probability
	mate_singlepoint_prob	Single point mating probability
	mate_only_prob	Mate only probability
	recur_only_prob	Recurrent only connection
	pop_size	Population Size
	dropoff_age	Number of generations before downward pressure is exerted
General Parameters	X_{face}	Number of faces applied per generation
	$Y_{nonface}$	Number of non-faces applied per generation
	false_positive_penalty	False positive penalty term
	false_negative_penalty	False negative penalty term
	α	Weight decay constant – input to output
	β	Weight decay constant – input to hidden
	γ	Weight decay constant – hidden to output
	δ	Weight decay constant – hidden to hidden
	No. of training iterations	
	starting_genome	Minimal topology or seeded
	bootstrap_max	Maximum non-face images added per bootstrapping iteration
	max_gen	Maximum number of training generations
	Target generalisation	

selecting these parameters [170]. For example, crossover is generally recommended to be high (> 0.8), as it encourages good solutions to breed and so create a potential for improved offspring. A large population maintains diversity, which is particularly useful if the search space is vast, as is the case with the chosen problem of face detection. The cost associated with a large population is a reduction in run-time performance. Diversity in the population can also be achieved using mutation, although good solutions can be lost if the value is set too high. Therefore mutation is generally set to a very low value (< 0.05). Observations derived from prior ANN-based face detection research can give an insight into the structure (number of neurons and connections) of the final evolved solution. For example, Rowley et al examined multiple networks ranging in size from 52 hidden neurons with 2,905 connections to 78 hidden neurons with 4,375 connections [84]. This also provides guidance for choosing the relative values for the add neuron mutation (*MUTATE_ADD_NODE_PROB*) and add connection mutation (*MUTATE_ADD_LINK_PROB*) parameters.

In the absence of an ideal solution with 100% recall and 100% precision under all circumstances, the question arises of whether recall or precision is more important. This is compounded by the fact that attempting to improve one, typically has a negative impact on the other. The considered opinion with regard to a final solution is that the answer to this question is application dependant. However, the author found that when evaluating the performance of a phenotype for suitability for normal mode operation (see Chapter 5), phenotypes with lower recall and higher precision (using figures generated from the validation dataset) gave improved performance compared to phenotypes with higher recall and lower precision. This could be attributed to the nature of the algorithm, which when in normal mode provides multiple opportunities to detect a face (neighbouring positions and neighbouring resolutions). Furthermore, because the amount of non-face test windows greatly out numbers the amount of face test windows for any typical image, even a small improvement in precision, will have a dramatic impact on reducing the number of false positives. Fewer detections also reduce the computational expense of the merging process (See Section 5.1.1). From these observations, the author chose to more heavily penalise false positives than false negatives during the training process.

Many of these rules of thumb can give a indication of sensible values for the parameters. However, further parameter optimisation is necessary, because not only do these parameters greatly influence the quality of the face detection performance, they also have a direct impact on the computational complexity and thus the power consumption of the final evolved solution. In particular

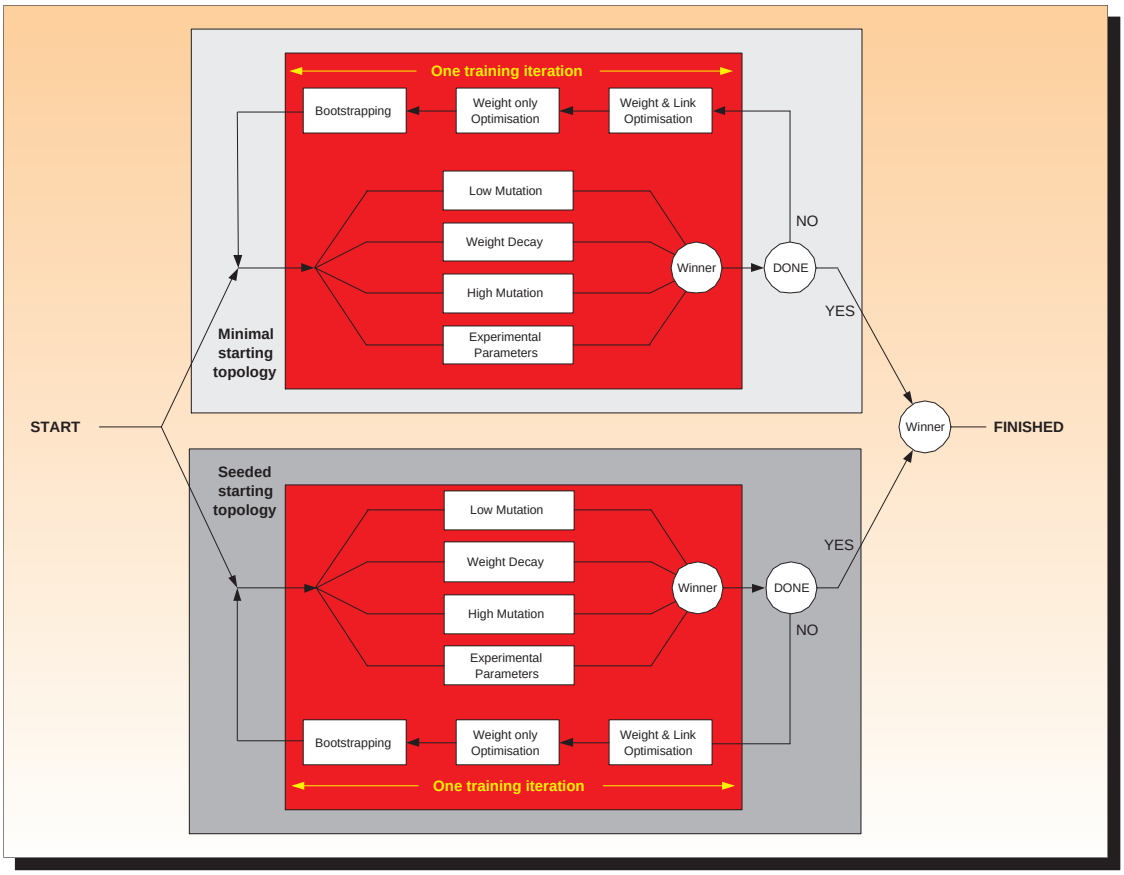


Figure 4.6: Proposed Evolutionary Training Strategy

the neuron mutation and connection mutation, which both add structure to the genomes, increases the computational expense of the algorithm. Unfortunately, there are too many parameters to exhaustively optimise, especially since a single training run can last in the order of days. Therefore training must be approached in a structured methodical manner. The block diagram in Fig. 4.6 shows the chosen methodical approach for optimising the large number of tunable parameters. Whereas previously Fig. 4.4 demonstrated the principal steps involved in one iteration of training, Fig. 4.6 shows the training at a higher level of abstraction with multiple iterations of multiple parallel training runs, which are used to investigate different parameter settings. As can be seen in Fig. 4.6 the training strategy diverges based on whether the initial starting topology for the evolutionary process is a seeded topology or a minimal structure topology (see Section 4.2). Apart from the different starting topologies, the steps involved in the training iterations for both the seeded and non-seeded training are identical.

The training strategy shown in Fig. 4.6 evaluates a number of parameters through parallel training runs. The winning population is selected from these (based upon the performances on the validation dataset) and provided the solution is still improving, a further iteration of parallel

training is carried out. Prior to this commencing, the extracted population undergoes two further optimising training runs. As can be seen in Fig. 4.6 the first optimisation is trained with the add neuron mutation having a value of zero (i.e. weight perturbation mutation and add link mutation only). The goal is to optimise the weight values and connections between the given neurons at this point. This is followed by training with no add neuron mutation or add link mutation. This focuses training on optimising the weight values for the given neurons and connections. The overall goal of these two training runs is to optimise the topology before adding further neuron structure. This is beneficial because as was observed by Stanley when structure is added to promising solutions in a lower dimensional space, the genome will already be in a promising position in the higher dimensional space [133]. The final step before commencing the next training iteration is to generate additional training data by bootstrapping this population. This work has used this bootstrapping process to only generate additional non-face data samples, though it could also be used to generate additional face samples in order to improve the recall rate.

The multiple parallel training runs allow exploration of GA and algorithm parameter settings. As the mutation parameters are responsible for creating distinctly different topological solutions, particular emphasis is placed on investigating suitable values for these parameters. Therefore two of the parallel runs are dedicated to investigating mutation parameter values (typically low values and high values respectively). A further training run is used to explore weight decay as a means of improving generalisation (See Section 4.2). The final parallel training run is used to make judicious adjustments to many of the parameters (e.g. investigating the effect of increasing X_{face} and $Y_{nonface}$ values as the training progresses). Of course further parallel training runs could also be used, however in this work 4 parallel runs offered the best trade off between parameter exploration and computational resources.

4.3.1 Non-Seeded Topology Training

This section describes the face detection training which was launched with an initial population spawned from a minimal topology (i.e. all the inputs are connected to a single output and there are no hidden neurons present). To increase the robustness of the final solution, multiple training iterations (i.e. steps 1 to 9 in Fig. 4.4 corresponds to one iteration) were run. This progressively increased the training dataset, whilst building upon the best topology and weight values from the previous iteration. Training iterations were carried out until 98% (chosen arbitrarily) precision was achieved. This corresponded to six iterations in the case of the non-seeded topology training.

To allow fair comparisons, the same number of iterations were also carried out for the seeded topology training. It should be noted that further training iterations are possible, which could further improve the generalisation ability of the evolved solution.

4.3.1.1 First Training Iteration

The parameters used for the first iteration of the parallel training are shown in Table 4.2. In each case training begins for the non-seeded topology with only randomised non-face data samples. Subsequent training runs gradually increase the non-face dataset through bootstrapping. For this and subsequent training runs unless otherwise stated, false_negative_penalty is 1.0, training was run for up to 4000 generations, the end point of the graphs is the last generation where a record fitness was found. The validation dataset contained 1594 faces (from the FERET dataset) and 64000 non-faces samples (taken from a scenery image). The parameters explored during the first training iteration could be broadly described as low mutation via *tr_ns_inc1_A*, *tr_ns_inc1_B* explores high mutation, *tr_ns_inc1_C* investigates low mutation with weight decay, whilst very low mutation with weight decay and a high drop_off age is explored in *tr_ns_inc1_D*. The resulting performance of each training run using the validation dataset is shown in Fig. 4.7. The most striking aspect from these graphs is the very low precision, which is caused by the high number of false positives. For example, the highest precision recorded was just over 20% (generation 418 in *tr_ns_inc1_B*), and whilst this phenotype correctly classified 1411 of 1594 faces, it also gave 5,487 false positives. However, considering that only randomised non-face data was used during training, this is understandable. Encouragingly, this phenotype only contains 4 hidden neurons and 324 connections, yet still managed to correctly classify 58513 of 64000 non-face in the validation dataset. To improve the precision, clearly more representative non-face data samples were required, this was achieved by bootstrapping (using generation 418 from training run *tr_ns_inc1_B*) an additional 1500 non-face data samples.

4.3.1.2 Second Training Iteration

The second training iteration increased the false positive penalty to 1.75 for each run. The performance of *tr_ns_inc1_B* provided the motivation for this. For this reason also, the lower valued mutation parameters (in *tr_ns_inc2_A* and *tr_ns_inc2_C*) were also increased. Similar to the previous training iteration, the first three training runs have the theme of low mutation, higher mutation and low mutation with weight decay (weight decay parameters reduced relative to the previous

run). The final training run doesn't include random faces and applies more non-faces than faces to each genome, the goal of which is to investigate whether stressing the importance of representative non-faces will improve the precision. All the parameters which have changed from the previous iteration are shown in bold in Table 4.3 (this is similar for subsequent training iterations). The results of the second training iteration are shown in Fig. 4.8. On this occasion, the training runs with the lower mutation values (*tr_ns_inc2_A* & *tr_ns_inc2_C*) considerably outperformed the high mutation training run (*tr_ns_inc2_B*). A recall rate of almost 87% with a precision of over 55% was recorded for generation 657 in *tr_ns_inc2_A*. This is a 30% increase in the precision compared to the previous training iteration and can be attributed to the bootstrapped non-face data samples. The associated network topology remains small with only 5 hidden neurons and 347 connections. It can be seen in Fig. 4.8 that *tr_ns_inc2_C* gave comparable results to *tr_ns_inc2_A*, which is to be expected due to the very small values for the weight decay constants. Unfortunately, the additional emphasis on the non-face data samples in *tr_ns_inc2_D* had a detrimental effect of the generalisation capabilities of the evolved solution. The optimising training runs (see Fig. 4.6) were carried out on generation 657 from *tr_ns_inc2_A*. This leads to a further improvement in recall (92%) and precision (57%).

4.3.1.3 Third Training Iteration

The third training iteration continued from the weight optimised *tr_ns_inc2_A* population. For each run the bootstrap data was increased to 3500 data images. 8000 randomised images were added to the bootstrap data for the total non-face training dataset for *tr_ns_inc3_D*. For the other three runs, 4000 randomised images were used. More randomised images were used in *tr_ns_inc3_D* to investigate any possible effects on the generalisation ability of the evolved phenotypes. With the exception of *tr_ns_inc3_D*, mutation values were the same as the previous training iteration. Lower mutation values, a higher false positive penalty and a higher drop-off age were used for training run *tr_ns_inc3_D*. The resultant evolution revealed that training run *tr_ns_inc3_A* gave the best performance on the validation dataset, with a recall rate 70% and precision of 86%. As can be seen in Fig. 4.9(a) this was achieved during generation 49 of *tr_ns_inc3_A*. The fittest phenotype in this generation used 5 hidden nodes and 348 connections. The two subsequent optimising training runs did not improve this performance. In each separate training run of the third training iteration, it can be seen in Fig. 4.7 that few new winning genomes were found after a brief initial period. This could potentially be explained by the effect of starting the evolution with a weight optimised

population. This coupled with low mutation values for add node and add link could have caused the search to get stuck in local minima at the start of the training iteration. Evidently the large values for the weight perturbation did not help during this training iteration. Therefore, the subsequent training iteration investigated alternative weight mutation perturbation levels (*weight_mut_power*) and also increased the mutation values for add node and add link. In light of these observations it is difficult to draw any conclusions about the experimental parameters used in *tr_ns_inc3_D*.

4.3.1.4 Fourth Training Iteration

Bootstrap data was increased to 6,000 samples for the fourth training iteration which used generation 49 from *tr_ns_inc3_A* as the starting population. Although, in the case of training run *tr_ns_inc4_D* rather than continue evolution, it was restarted from a minimal topology to investigate if a completely new evolution using the current dataset could improve performance. The structural mutation values ($mutate\_add\_node\_prob$ and $mutate\_add\_link\_prob$) were increased and alternative values were explored for the weight mutation power for the reasons outlined previously. The values for X_{face} & $Y_{nonface}$ were also increased so that more representative samples were applied to each genome within a population per generation. The value of the weight decay parameters were also increased in *tr_ns_inc4_C* relative to the training run *tr_ns_inc3_C*. After 1,500 generations the recall and precision was considerably below that achieved via the incremental evolution and the fittest genome at this point contained more hidden neurons (11) and connections (362). The fittest genome found during this training iteration occurred in the weight decay training run *tr_ns_inc4_C*. This was optimised further and a genome with a recall of almost 75% with a precision of 93% was found. This genome contained 9 hidden neurons and 352 connections.

4.3.1.5 Fifth Training Iteration

The fifth training iteration continued from generation 669 of *tr_ns_inc4_A_opt2*. Bootstrapped non-face data was increased to 10,000 samples for *tr_ns_inc5_C* and 9,000 samples for the other training runs. Due to the improvements observed in the previous training iteration, the weight mutation power was considerably decreased for each training run. Training runs *tr_ns_inc5_B*, *tr_ns_inc5_C* & *tr_ns_inc5_D* found few new fit genomes. In particular training run *tr_ns_inc5_B* performed poorly and this could be attributed to too high a value for the false positive penalty term on this occasion and too few representative training samples per generation. Fortunately, many new fit genomes were found in training *tr_ns_inc5_A*, including a genome in generation 1,378

which gave a recall rate of 78% with a precision of 93% when operating on the validation dataset. This genome contained 14 hidden neurons and 408 connections.

4.3.1.6 Sixth Training Iteration

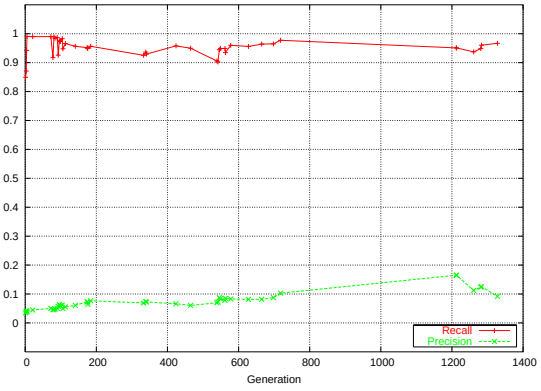
The sixth iteration continued from generation 1378 from *tr_ns_inc5_A*. As previously discussed, the best resulting performance occurred during generation 704 of *tr_ns_inc6_A* and gave a recall of over 71% with a precision of just under 97%. Although the recall rate is reduced relative to the previous training iteration, this still represents an improvement as the precision, which is more important has improved. This was further improved in the weight only mutation optimising training run, resulting in a genome which had a recall rate of 70.6% with a precision of 98%. This genome used 15 hidden neurons and 370 connections. Using unequal values for X_{face} and $Y_{nonface}$, a higher weight decay constant or increasing the drop-off age for training run *tr_ns_inc5_C* did not assist the evolutionary search during this training iteration. The performance of this population on the testing dataset are described further in Section 4.4.

4.3.2 Seeded Topology Training

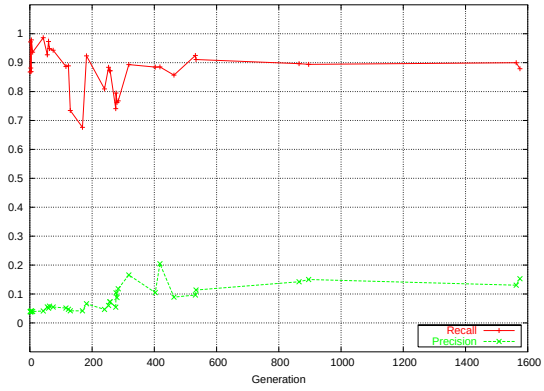
A common approach when using ANNs for face detection is to use a "semantic receptive field" type topology, whereby the hidden nodes are connected to inputs in order to create local receptive fields for semantic regions (eyes, nose, mouth etc) in the face [84]. Whilst it should be clear that NEAT provides the opportunity for discovering novel topologies, the question remains whether NEAT can be guided toward a more structured "semantic receptive field" type topology. Furthermore, would such a topology outperform a topology generated from a conventional NEAT evolution? It is difficult to predict the answer to these questions. There is a possibility that any potential benefit of a topology seeded with semantic receptive fields may be offset by the challenge for the evolutionary search in finding a solution in a search space with a larger initial size. Therefore, this section details an alternative training phase which attempts to allow experimental insight into the outlined questions. This training differs from the previous training in that a seeded topology is used in the initial training iteration. The seeded topology has two hidden neurons present, which are connected to the eyes and mouth regions respectively (see Section 4.2 for more details). A discussion regarding the performance and characteristics of the non-seeded and seeded training is provided in Section 4.4.

Table 4.2: Parameters for iteration 1 of non-seeded starting topology training run

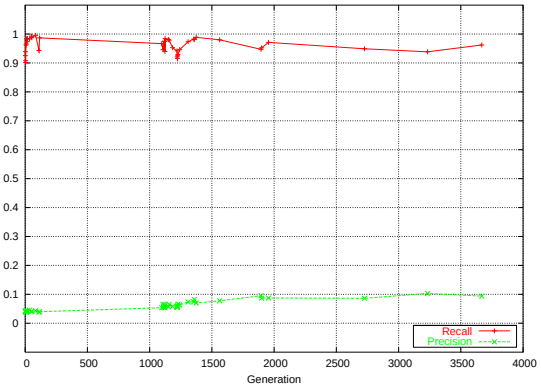
Parameter Value	Training Runs			
	tr_ns_inc1_A	*tr_ns_inc1_B*	*tr_ns_inc1_C*	*tr_ns_inc1_D*
Population	150	150	150	150
Total faces	1884	1884	1884	1884
Total random nonfaces	4000	4000	4000	4000
Total bootstrapped nonfaces	0	0	0	0
X_{face}	200	100	200	300
$Y_{nonface}$	200	100	200	300
false_positive_penalty	1.50	1.75	1.25	1.00
weigh_mut_power	2.5	2.5	2.5	2.5
mutate_add_node_prob	0.0001	0.001	0.0001	0.00001
mutate_add_link_prob	0.005	0.05	0.005	0.001
α	0	0	0.00000075	0.00001
β	0	0	0.00000001	0.00001
γ	0	0	0.00000001	0.00001
δ	0	0	0.00000001	0.00001
dropoff_age	100	50	100	200



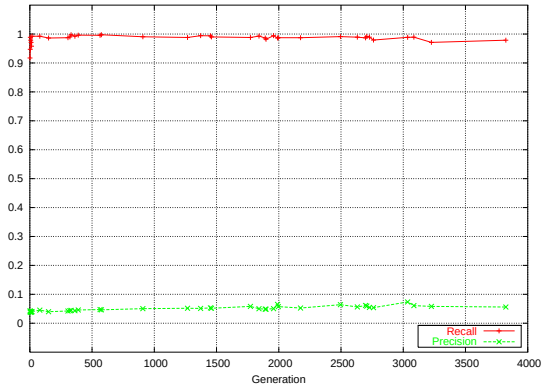
(a) *tr_ns_inc1_A* - validation dataset



(b) *tr_ns_inc1_B* - validation dataset



(c) *tr_ns_inc1_C* - validation dataset

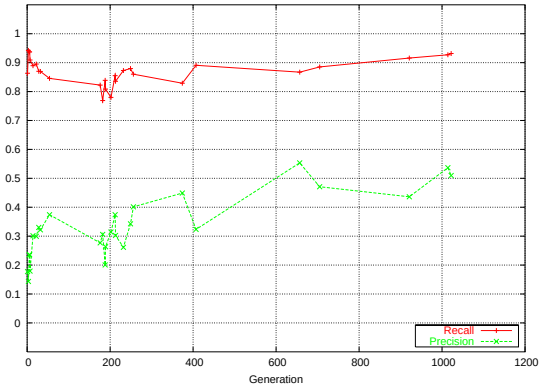


(d) *tr_ns_inc1_D* - validation dataset

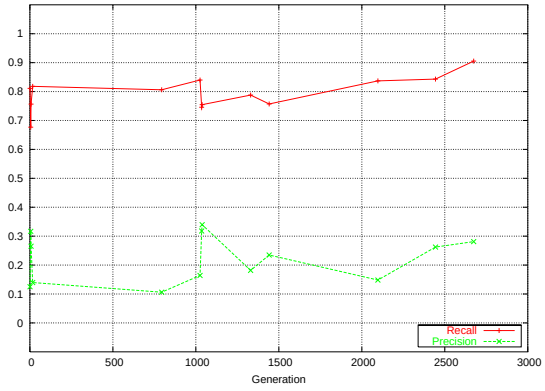
Figure 4.7: Non-seeded topology training run – iteration 1

Table 4.3: Parameters for iteration 2 of non-seeded starting topology training run

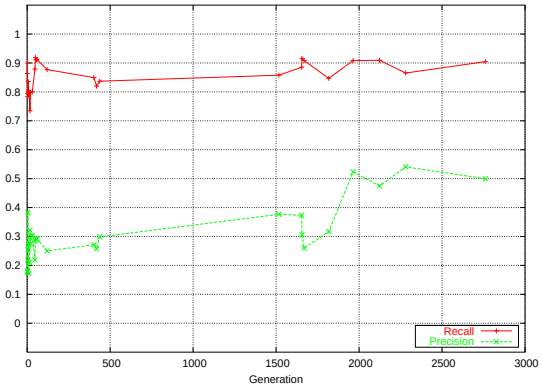
Parameter Value	Training Runs			
	tr_ns_inc2_A	*tr_ns_inc2_B*	*tr_ns_inc2_C*	*tr_ns_inc2_D*
Population	150	150	150	150
Total faces	1884	1884	1884	1884
Total random nonfaces	4000	4000	4000	**0**
Total bootstrapped nonfaces	**1500**	**1500**	**1500**	**1500**
X_{face}	200	100	200	**150**
$Y_{nonface}$	200	100	200	**200**
false_positive_penalty	**1.75**	1.75	**1.75**	**1.75**
weigh_mut_power	2.5	2.5	2.5	2.5
mutate_add_node_prob	**0.0005**	0.001	**0.0005**	**0.0005**
mutate_add_link_prob	**0.01**	0.05	**0.01**	**0.01**
α	0	0	**0.00000001**	**0**
β	0	0	0.00000001	**0**
γ	0	0	0.00000001	**0**
δ	0	0	0.00000001	**0**
dropoff_age	100	50	100	**75**



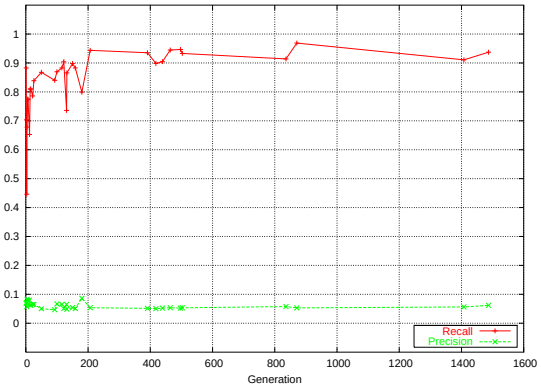
(a) *tr_ns_inc2_A* - validation dataset



(b) *tr_ns_inc2_B* - validation dataset



(c) *tr_ns_inc2_C* - validation dataset

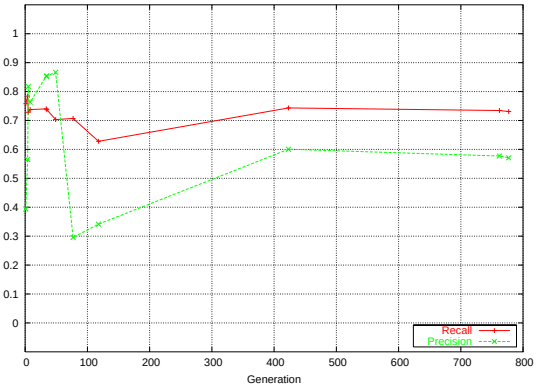


(d) *tr_ns_inc2_D* - validation dataset

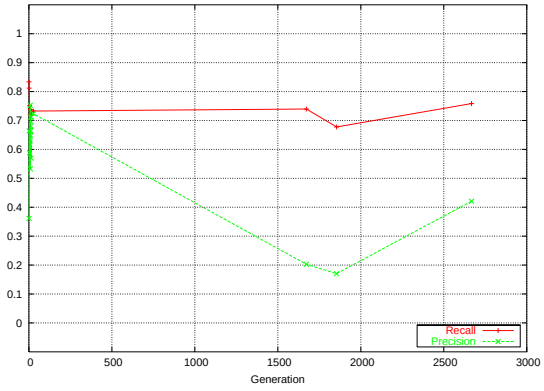
Figure 4.8: Non-seeded topology training run – iteration 2

Table 4.4: Parameters for iteration 3 of non-seeded starting topology training run

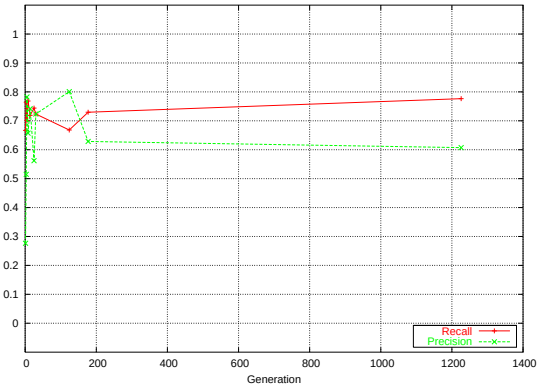
Parameter Value	Training Runs			
	tr_ns_inc3_A	*tr_ns_inc3_B*	*tr_ns_inc3_C*	*tr_ns_inc3_D*
Population	150	150	150	150
Total faces	1884	1884	1884	1884
Total random nonfaces	4000	4000	4000	**8000**
Total bootstrapped nonfaces	**3500**	**3500**	**3500**	**3500**
X_{face}	200	**125**	200	150
$Y_{nonface}$	200	**125**	**200**	150
false_positive_penalty	1.75	1.75	1.75	2.00
weigh_mut_power	2.5	2.5	2.5	2.5
mutate_add_node_prob	0.0005	0.001	0.0005	**0.00025**
mutate_add_link_prob	0.01	0.05	0.01	**0.005**
α	0	0	**0.00000100**	0
β	0	0	**0.00000010**	0
γ	0	0	**0.00000010**	0
δ	0	0	**0.00000010**	0
dropoff_age	100	50	100	**120**



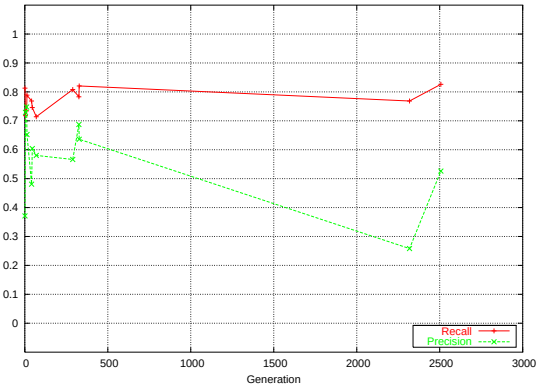
(a) *tr_ns_inc3_A* - validation dataset



(b) *tr_ns_inc3_B* - validation dataset



(c) *tr_ns_inc3_C* - validation dataset

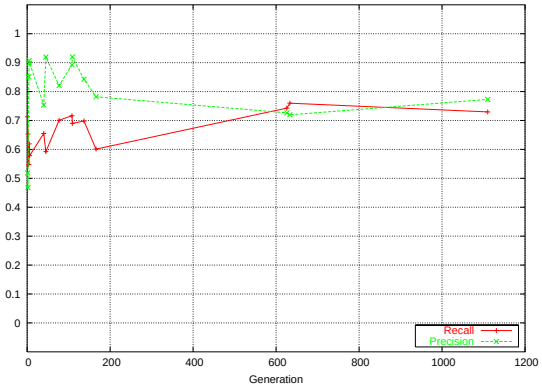


(d) *tr_ns_inc3_D* - validation dataset

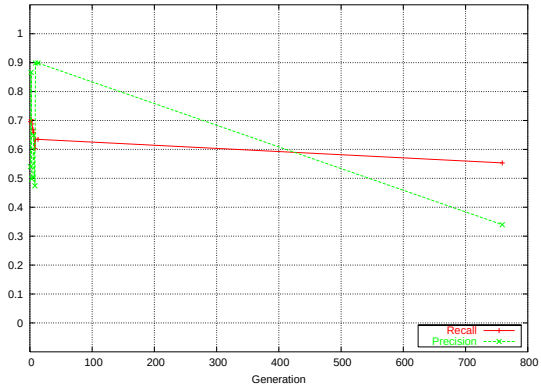
Figure 4.9: Non-seeded topology training run – iteration 3

Table 4.5: Parameters for iteration 4 of non-seeded starting topology training run

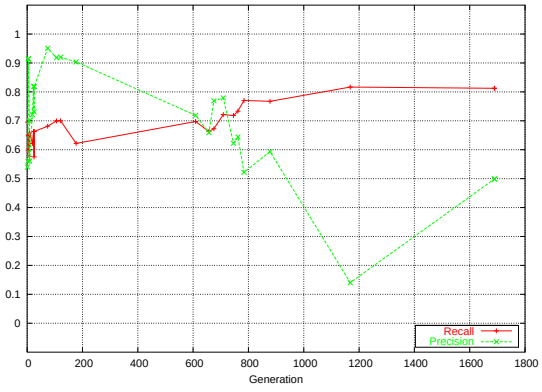
Parameter Value	Training Runs			
	tr_ns_inc4_A	*tr_ns_inc4_B*	*tr_ns_inc4_C*	*tr_ns_inc4_D*
Population	150	150	150	150
Total faces	1884	1884	1884	1884
Total random nonfaces	4000	4000	4000	**4000**
Total bootstrapped nonfaces	**6000**	**6000**	**6000**	**6000**
X_{face}	**250**	**200**	200	**1000**
$Y_{nonface}$	**250**	**200**	200	**1000**
false_positive_penalty	1.75	1.75	1.75	1.75
weigh_mut_power	**1.5**	**3.5**	2.5	**2.0**
mutate_add_node_prob	**0.005**	**0.02**	**0.001**	**0.001**
mutate_add_link_prob	0.01	**0.08**	0.01	**0.01**
α	0	0	**0.0000100**	0
β	0	0	**0.0000010**	0
γ	0	0	**0.0000010**	0
δ	0	0	**0.0000010**	0
dropoff_age	100	**100**	100	**40**



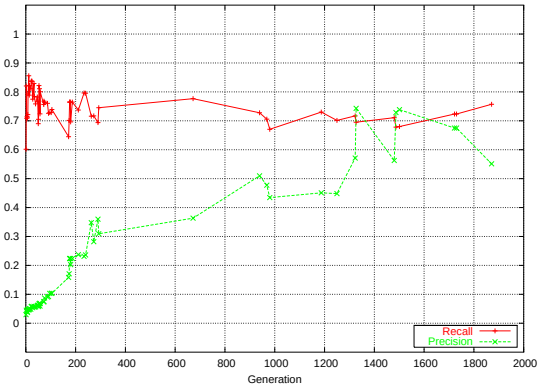
(a) *tr_ns_inc4_A* - validation dataset



(b) *tr_ns_inc4_B* - validation dataset



(c) *tr_ns_inc4_C* - validation dataset

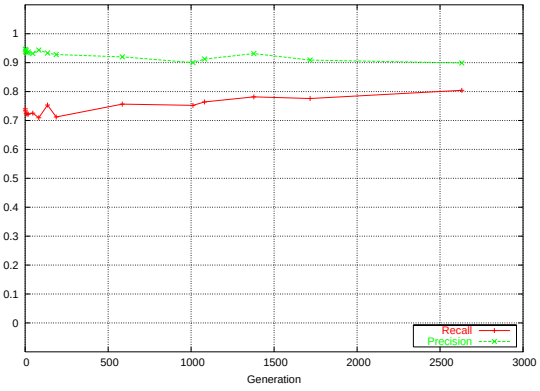


(d) *tr_ns_inc4_D* - validation dataset

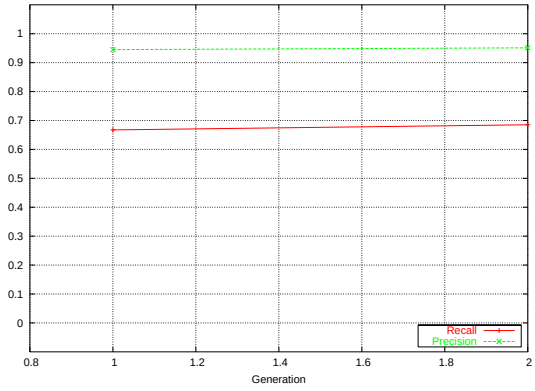
Figure 4.10: Non-seeded topology training run – iteration 4

Table 4.6: Parameters for iteration 5 of non-seeded starting topology training run

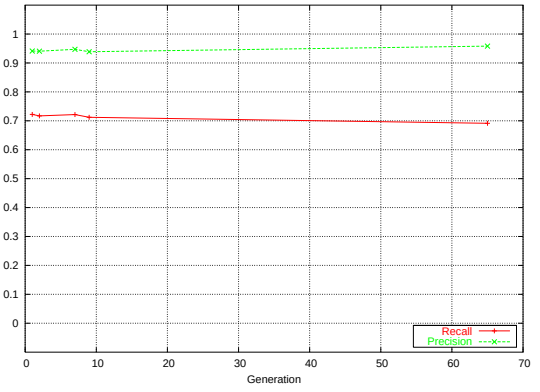
Parameter Value	Training Runs			
	tr_ns_inc5_A	*tr_ns_inc5_B*	*tr_ns_inc5_C*	*tr_ns_inc5_D*
Population	150	150	150	150
Total faces	1884	1884	1884	1884
Total random nonfaces	4000	4000	**1000**	**0**
Total bootstrapped nonfaces	**9000**	**9000**	**10000**	**9000**
X_{face}	250	**100**	200	**200**
$Y_{nonface}$	250	**100**	200	**200**
false_positive_penalty	1.75	**2.00**	1.75	1.75
weigh_mut_power	**0.1**	**0.25**	**0.05**	**0.5**
mutate_add_node_prob	**0.0005**	0.02	0.001	**0.01**
mutate_add_link_prob	0.01	0.08	0.01	0.01
α	0	0	0.0000100	0
β	0	0	0.0000010	0
γ	0	0	0.0000010	0
δ	0	0	0.0000010	0
dropoff_age	100	100	100	**150**



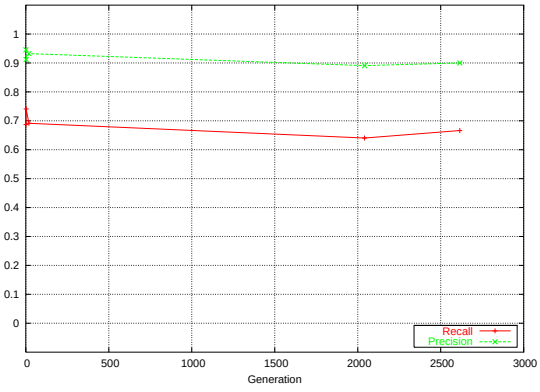
(a) *tr_ns_inc5_A* - validation dataset



(b) *tr_ns_inc5_B* - validation dataset



(c) *tr_ns_inc5_C* - validation dataset

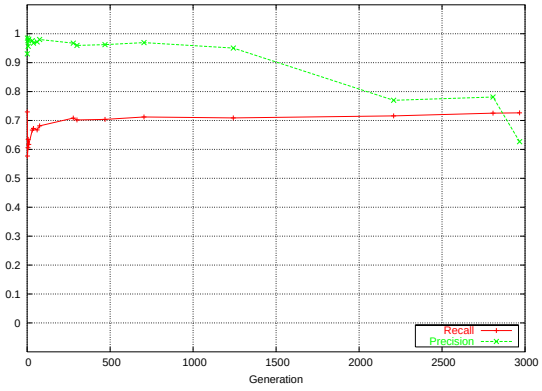


(d) *tr_ns_inc5_D* - validation dataset

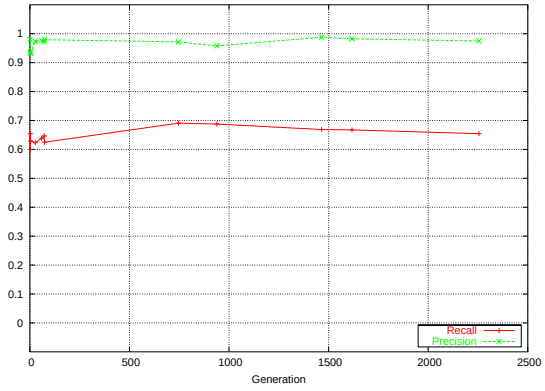
Figure 4.11: Non-seeded topology training run – iteration 5

Table 4.7: Parameters for iteration 6 of non-seeded starting topology training run

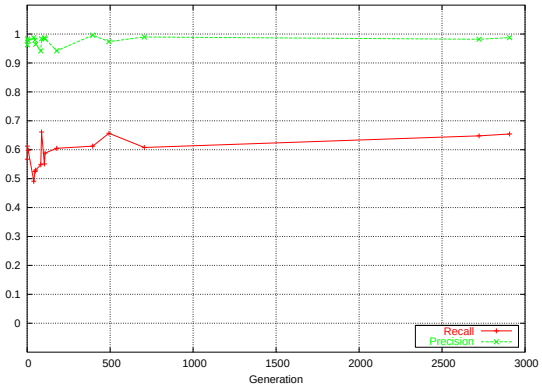
Parameter Value	Training Runs			
	tr_ns_inc6_A	*tr_ns_inc6_B*	*tr_ns_inc6_C*	*tr_ns_inc6_D*
Population	150	150	150	150
Total faces	1884	1884	1884	1884
Total random nonfaces	**1000**	4000	**8000**	0
Total bootstrapped nonfaces	**12000**	**12000**	**12000**	**13800**
X_{face}	250	**200**	**100**	200
$Y_{nonface}$	250	**300**	**300**	**300**
false_positive_penalty	1.75	**1.6**	1.75	1.75
weigh_mut_power	0.1	**0.05**	**0.25**	0.5
mutate_add_node_prob	0.0005	**0.01**	**0.00005**	**0.001**
mutate_add_link_prob	0.01	**0.04**	**0.001**	**0.001**
α	0	0	**0.0000200**	0
β	0	0	0.0000010	0
γ	0	0	0.0000010	0
δ	0	0	0.0000010	0
dropoff_age	100	100	**200**	150



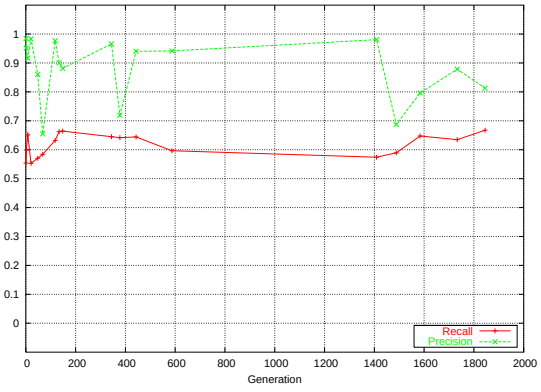
(a) *tr_ns_inc6_A* - validation dataset



(b) *tr_ns_inc6_B* - validation dataset



(c) *tr_ns_inc6_C* - validation dataset



(d) *tr_ns_inc6_D* - validation dataset

Figure 4.12: Non-seeded topology training run – iteration 6

4.3.2.1 First Training Iteration

The parameters used for each training run in the first iteration of the seeded training are shown in Table 4.8. With the exception of the initial starting topology, this training phase is identical to that presented in Fig. 4.4. With regard to the mutation operators, the author initially felt that creating an *add group of links* type mutation operator similar to that proposed by Wiegand et al could be beneficial [134]. With such a mutation operator instead of a single connection being created between an input neuron and a hidden neuron, links from a single hidden neuron are created to multiple input neurons with the goal of grouping pixels into semantic units . However it is not obvious how many connections should be created or even if multiple shaped connection regions (e.g. circular, square, rectangular) would be more appropriate. Therefore, the *add group of links* type mutation operator proposed by Wiegand et al was not used. As an alternative, higher probability for *add link* mutation was investigated for all seeded training runs. This was chosen so as to create more connections to semantically important groups of pixels, although it also increased the number of connections between hidden neurons. However, the increased mutation probability lead to the search becoming less structured, which had a detrimental effect of the overall progress of the training. Therefore, a conventional NEAT strategy was employed to handle connection mutation.

In the first iteration as there is already hidden neurons and a considerable number of links present in the starting topology, the principal goal was to optimise the weights of those connections before adding further structure during subsequent training. As a result the value of the *add node* mutation is set to zero. Similarly, the *add link* mutation is set to low values (tr_seeded_inc1_A and tr_seeded_inc1_B) or zero (tr_seeded_inc1_C & tr_ns_inc1_D) values are set. The number of faces and non-faces applied to each genome per generation ranged from 100 to 300. In each case 4000 randomised nonface images were used. Due to the observations regarding suitable ranges for the weight mutation power in the non-seeded training, this initial training iteration had considerably lower values for this parameter. Similarly, a common value of 1.75 was used for the false positive penalty, based upon the performance in the non-seeded training. Non-zero weight decay values was used in tr_seeded_inc1_C and tr_seeded_inc1_D.

The recall and precision of training runs tr_seeded_inc1_A and tr_seeded_inc1_B were better than training runs tr_seeded_inc1_C and tr_seeded_inc1_D, this could be attributed to the non-zero mutation in tr_seeded_inc1_A and tr_seeded_inc1_B along with zero valued weight decay constants. The fittest genome found during this training iteration occurred in generation 48 in training run

tr_seeded_inc1_A (see Fig.4.13(a)). This genome gave an 88% recall and 15% precision when using the validation dataset. Similar to the non-seeded training the precision is very low. This is understandable due to the use of only random non-face data. As well as having the three initial hidden neurons, the winning genome contained 479 links. As no add node mutation was used, no further optimisation was carried out on this generation.

4.3.2.2 Second Training Iteration

To improve the precision from the first training iteration 1000 bootstrapped non-face images were added to each training run in the second iteration. In addition, the add node mutation values were changed to nonzero values and the add link mutation values were increased. For experimentation purposes, the add node mutation was set to a higher value than the add link mutation in tr_seeded_inc2_D. The values of X_{face} and $Y_{nonface}$ were also changed to a consistent value of 200 to make comparison easier in this training iteration. Lower weight decay constants were explored in tr_seeded_inc2_D. With these changes the fittest phenotype found in this iteration occurred in generation 97 of *tr_seeded_inc2_D* and gave a recall of over 77% with a precision of just under 50% (see Fig.4.15(d)). As this phenotype did not contain additional structure (i.e. three hidden nodes and 479 connections) over the fittest phenotype in the previous iteration, the contributing factors in the improved performance were the additional bootstrapped data coupled with mutations of the weight values. The two subsequent optimising training runs did not improve the performance.

4.3.2.3 Third Training Iteration

An increase in the total quantity of non-face training samples was the principal change in the third training iteration. Bootstrapped non-face data was increased in all training runs, although, reduced randomised samples were explored in three of the parallel training runs. Considerably larger γ and δ weight decay constants were also investigated in tr_seeded_inc3_C. The resultant fittest phenotype found in this training iteration occurred in generation 918 of *tr_seeded_inc3_A* (see Fig.4.15(a)), achieving a recall rate of over 86% with a precision of just under 44%. As there were few changes to the parameters in this training run relative to the previous iteration, it is reasonable to conclude that this improvement was directly related to the additional non-face data and extra evolved structure. The performance was further improved to 83% recall with a precision of 49% using a weight only optimisation training run. This phenotype contained ten hidden nodes

and 503 connections.

4.3.2.4 Fourth Training Iteration

The fourth iteration increased the amount of bootstrapped data to 7000 non-face samples for each of the parallel training runs. Based upon the observations and general trends in the previous training iteration, the add node mutation probability value in both *tr_seeded_inc4_B* and *tr_seeded_inc4_D* was reduced. For similar reasons the weight mutation value in these two training runs was also reduced. Using a lowered penalty term for the false positives was explored in tr_seeded_inc4_D, with the goal of investigating whether the recall rate could be improved. The fittest phenotype found in this iteration occurred during generation 53 of *tr_seeded_inc4_D* and gave a recall rate of over 71% with a precision of just under 69% (see Fig.4.16(d)). The two subsequent optimising training runs improved this performance to 77% recall rate and over 70% precision. The phenotype contained no additional structure over the previous iteration (i.e. 10 hidden nodes and 503 connections). This is not surprising since the phenotype occurred early in the training iteration. The performance improvement could be attributed to a combination of weight optimisation, the reduction in false positive penalty term and the additional non-face training data.

4.3.2.5 Fifth Training Iteration

The bootstrapped non-face data was increased to 10,000 samples for the fifth training iteration. This was coupled with a small increase in the number of randomised samples. Add node and add link mutation probability values were maintained. Based upon the performances in prior training iterations the value of weight mutation was lowered. In addition, a further reduction in the false positive penalty term was explored in *tr_seeded_inc5_C*. However, as Fig. 4.17(c) shows, precision was heavily traded off for improved recall, which is undesirable in the chosen application. The best performance was recorded in generation 208 of *tr_seeded_inc5_B* and this phenotype gave a recall of 67% with a precision of 89% (see Fig.4.17(b)). Subsequent optimising training runs did not improve this performance. The phenotype contained 15 hidden nodes and 522 connections.

4.3.2.6 Sixth Training Iteration

In the sixth training iteration, the bootstrapped data was increased to 13660 images and the random data was also increased to 7000 images. The X_{face} and $Y_{nonface}$ values were also increased to 400. The mutation values used in the previous iteration were maintained, with the exception

that *tr_seeded_inc6_D* used a larger false positive penalty term in order to explore if this could improve the precision of the detection results. Weight decay constants were also adjusted in *tr_seeded_inc6_C* and *tr_seeded_inc6_D* relative to the previous training iteration. The best performing phenotype was found in generation 416 of *tr_seeded_inc6_A* (see Fig.4.18(a)). This phenotype gave a recall of 66% with a precision of 96%. Further optimising training runs marginally improved this performance to 67% recall and 96% precision. The phenotype was actually smaller than the best phenotype from the previous training iteration having only 13 hidden nodes and 520 connections.

4.4 Conclusions on Training

The non-seeded and seeded training schemes were each run for six iterations. At the end of these iterations a comparable number of non-face images had been added through a bootstrapping process. In addition, the number of face images used in each was identical. The generalisation ability of interim solutions for both schemes was evaluated using an identical validation dataset. Overall, this provides a platform for a fair comparison and evaluation of the merits of both schemes, whilst acknowledging some aspects of inherent randomness of the training algorithm (e.g. random non-face training samples, GA operation). The best performing phenotype from the non-seeded training achieved a recall rate of 70.6% and a precision of 98%. This compares to 67% recall and 96% precision for the best performing phenotype in the seeded training scheme. Furthermore, the non-seeded trained phenotype contained 370 connections as opposed to the 520 connections in the seeded trained phenotype. Although, the non-seeded phenotype contained two fewer neurons (13 as opposed to 15). Therefore, it is reasonable to claim that the non-seeded training scheme generated a better performing solution with a smaller overall topology. The poorer performance of the seeded training could be attributed to the fact that the initial search space was considerably larger and therefore harder to optimise. Both evolved solutions compare favourably to the ANN based face detection scheme of Rowley et al, which had a minimum size of 52 neurons and 2905 connections [84]. As will be shown in the benchmarking in the next chapter, the trained solution gives good performance for evaluation datasets that are representative of what is likely to be generated on a mobile device. The smaller topologies generated from the proposed training scheme are particularly attractive for a mobile device, as it will lead to a computational cost reduction. This in turn will allow improvements in the run-time performance and power consumption when

evaluating the proposed topology compared to the topology proposed by Rowley et al [84].

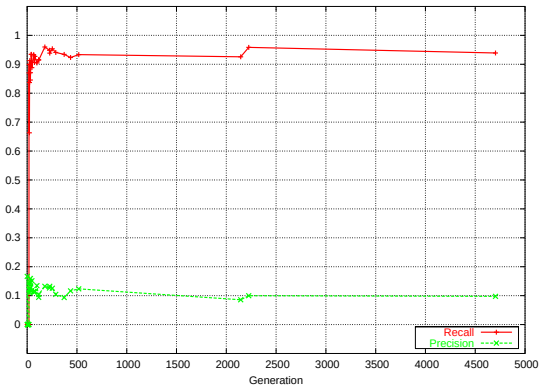
An extensive exploration of GA and algorithm parameters was carried out (see Section 4.2). In some cases low mutation worked best and on other occasions high mutation gave better results. Similar behaviour was noted for the majority of the parameters including the weight decay constants. As such, the author concludes that its highly unlikely that a fixed set of ideal parameters exist. Rather the parameters ideally should be dynamic to respond to the progress of the evolutionary search. For example, as the topology grows in size its quite likely the relative values of mutation should change in response to the existence of a greater number of local minima. In some respects using multiple iterations of trainings where different parameters are explored, could be considered to be a manual implementation of this concept. Furthermore, using multiple parallel training runs gives greater credence when extracting general trends from the training runs. The incremental approach to training was also shown to find a smaller topology in the fourth iteration of the non-seeded training relative to completely restarting the training process (see Fig. 4.10(d)). This could be explained by the finer granularity of the parameter exploration. A further contributing factor to the improved performance is that the problem becomes progressively more difficult in the incremental training approach. This may actually help to ease the task of optimising the earlier topologies and place genomes in promising locations in the solution subspace before the dataset becomes more difficult due to the addition of bootstrapping data. This is similar to the strategy successfully adopted by Stanley, in which he evolved solutions for smaller problems before building on these for more complex problems, such as his demonstration of using NEAT to play the popular board game GO [156].

4.5 Future Work

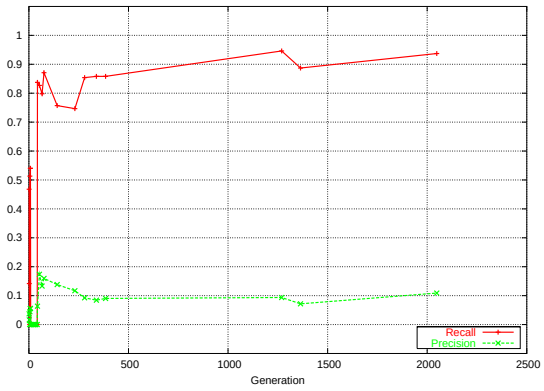
The proposed training scheme can be improved in terms of speed optimisations leading to improvements in the detection performance (i.e. recall & precision) and increasing the robustness of the algorithm (i.e. allowing detection with different face orientations). From these observations the following were identified as potential ways of improving the algorithm: automatic input/feature selection, using alternative low-level features, cascaded classifier operation and detection of side profile and rotated faces A detailed description is presented in the following sections.

Table 4.8: Parameters for iteration 1 of seeded starting topology training run

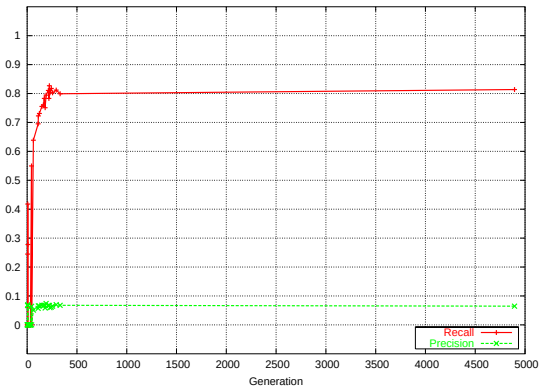
Parameter Value	Training Runs			
	tr_seeded_inc1_A	*tr_seeded_inc1_B*	*tr_seeded_inc1_C*	*tr_seeded_inc1_D*
Population	150	150	150	150
Total faces	1884	1884	1884	1884
Total random nonfaces	4000	4000	4000	4000
Total bootstrapped nonfaces	0	0	0	0
X_{face}	200	100	200	300
$Y_{nonface}$	200	100	200	300
false_positive_penalty	1.75	1.75	1.75	1.75
weigh_mut_power	0.25	1.5	0.5	1.0
mutate_add_node_prob	0.0	0.0	0.0	0.0
mutate_add_link_prob	0.005	0.05	0.0	0.0
α	0	0	0.00000075	0.00001
β	0	0	0.00000001	0.00001
γ	0	0	0.00000001	0.00001
δ	0	0	0.00000001	0.00001
dropoff_age	100	50	100	200



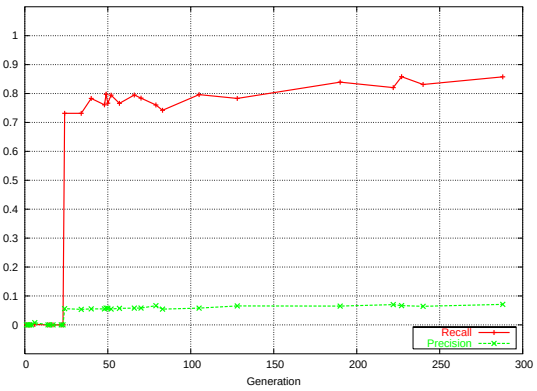
(a) *tr_seeded_inc1_A* - validation dataset



(b) *tr_seeded_inc1_B* - validation dataset



(c) *tr_seeded_inc1_C* - validation dataset

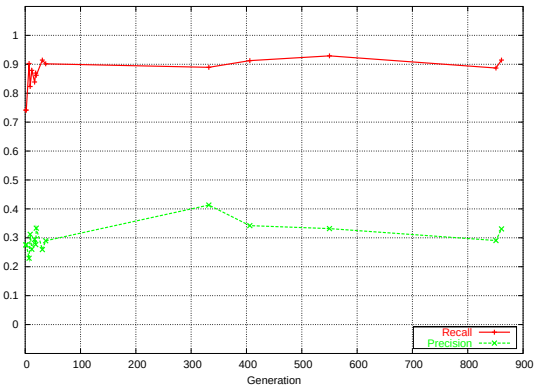


(d) *tr_seeded_inc1_D* - validation dataset

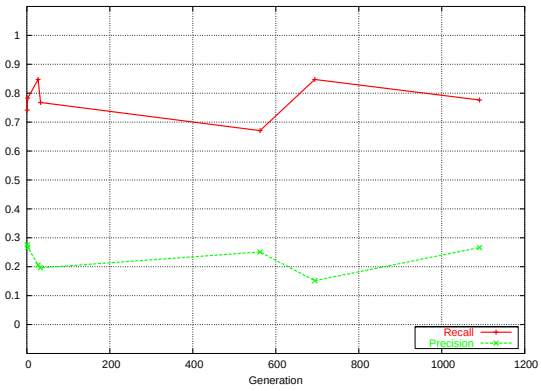
Figure 4.13: Seeded topology training run – iteration 1

Table 4.9: Parameters for iteration 2 of seeded starting topology training run

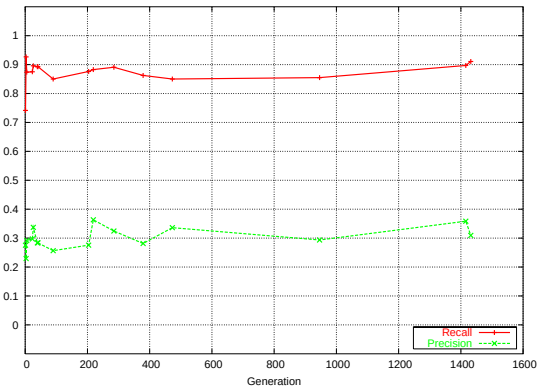
Parameter Value	Training Runs			
	tr_seeded_inc2_A	*tr_seeded_inc2_B*	*tr_seeded_inc2_C*	*tr_seeded_inc2_D*
Population	150	150	150	150
Total faces	1884	1884	1884	1884
Total random nonfaces	4000	4000	4000	4000
Total bootstrapped nonfaces	**1000**	**1000**	**1000**	**1000**
X_{face}	200	**200**	200	**200**
$Y_{nonface}$	200	**200**	200	**200**
false_positive_penalty	1.75	1.75	1.75	1.75
weigh_mut_power	0.25	1.5	0.5	1.0
mutate_add_node_prob	**0.0005**	**0.025**	**0.001**	**0.01**
mutate_add_link_prob	**0.001**	0.05	**0.01**	**0.005**
α	0	0	0.00000075	**0.0000001**
β	0	0	0.00000001	**0.0000001**
γ	0	0	0.00000001	**0.0000001**
δ	0	0	0.00000001	**0.0000001**
dropoff_age	100	50	100	200



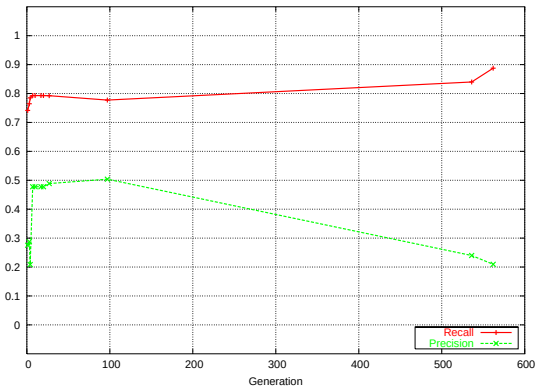
(a) *tr_seeded_inc2_A* - validation dataset



(b) *tr_seeded_inc2_B* - validation dataset



(c) *tr_seeded_inc2_C* - validation dataset

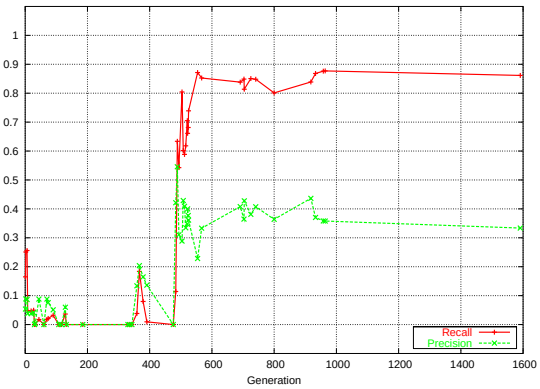


(d) *tr_seeded_inc2_D* - validation dataset

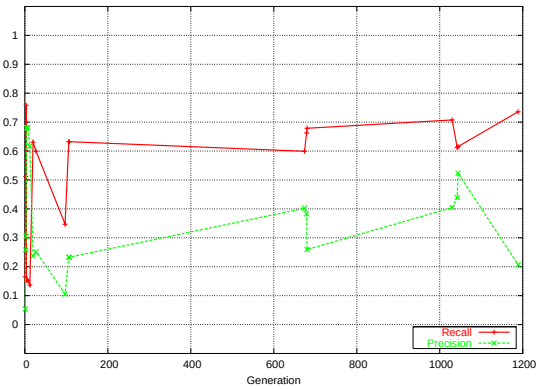
Figure 4.14: Seeded topology training run – iteration 2

Table 4.10: Parameters for iteration 3 of seeded starting topology training run

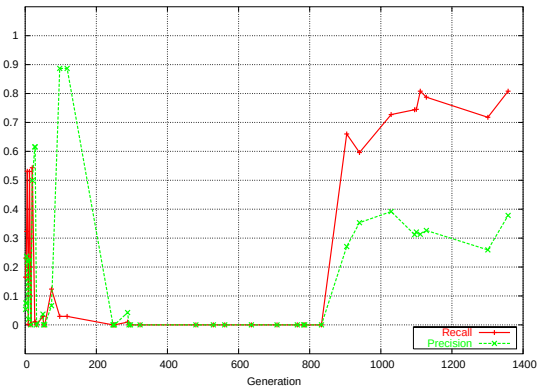
Parameter Value	Training Runs			
	tr_seeded_inc3_A	*tr_seeded_inc3_B*	*tr_seeded_inc3_C*	*tr_seeded_inc3_D*
Population	150	150	150	150
Total faces	1884	1884	1884	1884
Total random nonfaces	4000	**3000**	**3000**	**3000**
Total bootstrapped nonfaces	**3500**	**3000**	**3000**	**3000**
X_{face}	200	**300**	**300**	**300**
$Y_{nonface}$	200	**300**	**300**	**300**
false_positive_penalty	1.75	1.75	1.75	1.75
weigh_mut_power	0.25	1.5	0.5	1.0
mutate_add_node_prob	0.0005	0.025	0.001	0.01
mutate_add_link_prob	0.001	0.05	0.01	0.005
α	0	0	**0.0000005**	0.0000001
β	0	0	**0.0000005**	0.0000001
γ	0	0	**0.000005**	0.0000001
δ	0	0	**0.000005**	0.0000001
dropoff_age	100	50	100	200



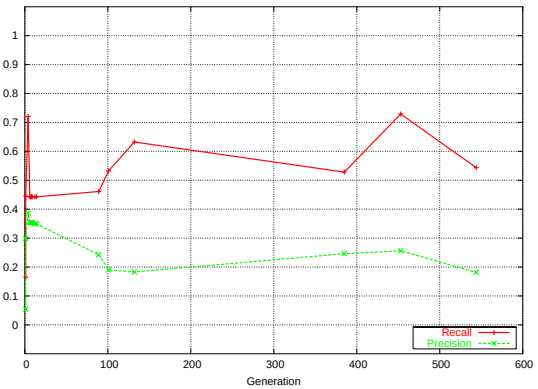
(a) *tr_seeded_inc3_A* - validation dataset



(b) *tr_seeded_inc3_B* - validation dataset



(c) *tr_seeded_inc3_C* - validation dataset

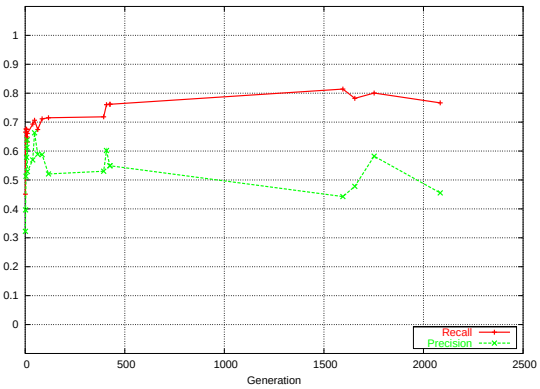


(d) *tr_seeded_inc3_D* - validation dataset

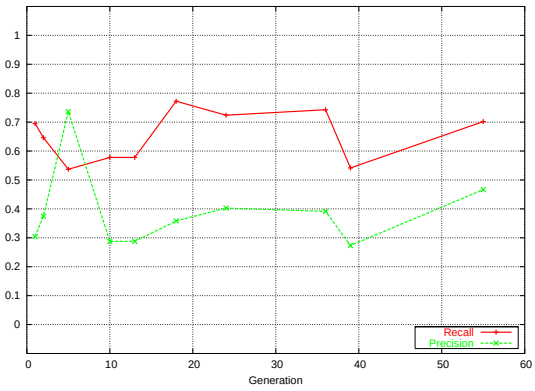
Figure 4.15: Seeded topology training run – iteration 3

Table 4.11: Parameters for iteration 4 of seeded starting topology training run

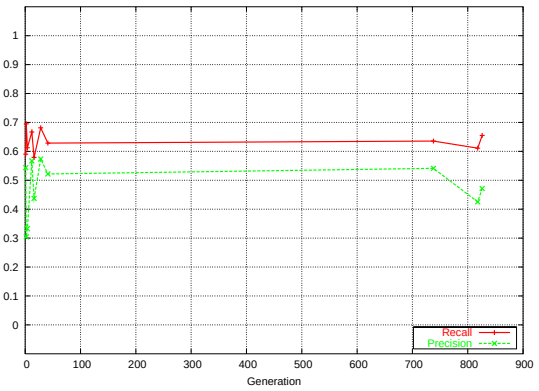
Parameter Value	Training Runs			
	tr_seeded_inc4_A	*tr_seeded_inc4_B*	*tr_seeded_inc4_C*	*tr_seeded_inc4_D*
Population	150	150	150	150
Total faces	1884	1884	1884	1884
Total random nonfaces	**3000**	3000	3000	3000
Total bootstrapped nonfaces	**7000**	**7000**	**7000**	**7000**
X_{face}	300	300	300	300
$Y_{nonface}$	300	300	300	300
false_positive_penalty	1.75	1.75	1.75	**1.5**
weigh_mut_power	0.25	**1.0**	0.5	**0.1**
mutate_add_node_prob	0.0005	**0.005**	0.001	**0.0025**
mutate_add_link_prob	0.001	0.05	0.01	0.005
α	0	0	0.0000005	0.0000001
β	0	0	0.0000005	0.0000001
γ	0	0	0.000005	0.0000001
δ	0	0	0.000005	0.0000001
dropoff_age	100	50	100	200



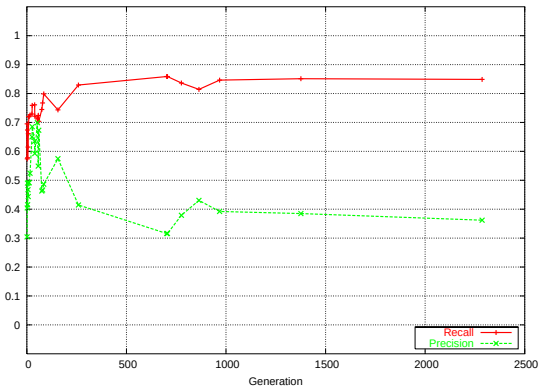
(a) *tr_seeded_inc4_A* - validation dataset



(b) *tr_seeded_inc4_B* - validation dataset



(c) *tr_seeded_inc4_C* - validation dataset

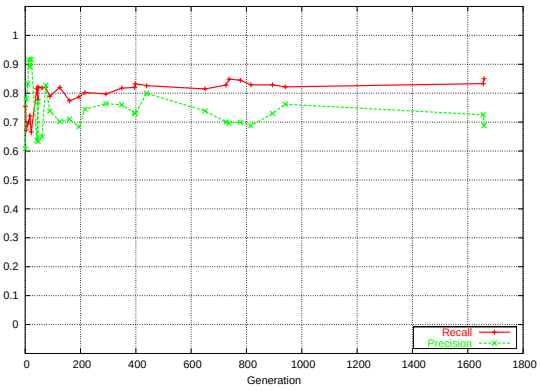


(d) *tr_seeded_inc4_D* - validation dataset

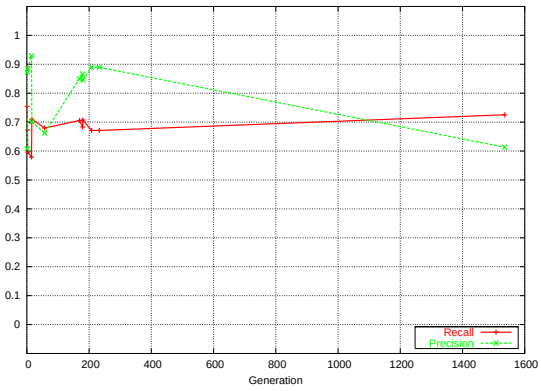
Figure 4.16: Seeded topology training run – iteration 4

Table 4.12: Parameters for iteration 5 of seeded starting topology training run

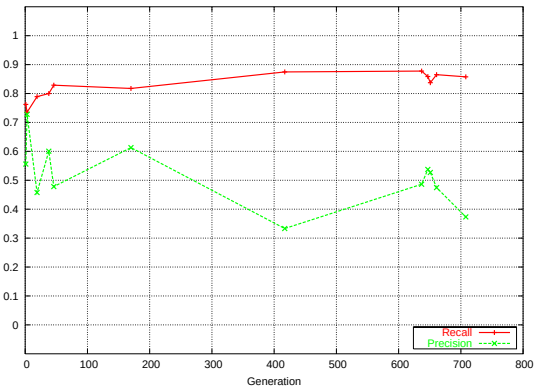
Parameter Value	Training Runs			
	tr_seeded_inc5_A	*tr_seeded_inc5_B*	*tr_seeded_inc5_C*	*tr_seeded_inc5_D*
Population	150	150	150	150
Total faces	1884	1884	1884	1884
Total random nonfaces	**5000**	**5000**	**5000**	**5000**
Total bootstrapped nonfaces	**10000**	**10000**	**10000**	**10000**
X_{face}	300	300	300	300
$Y_{nonface}$	300	300	300	300
false_positive_penalty	1.75	1.75	**1.25**	1.5
weigh_mut_power	**0.01**	**0.5**	**0.25**	0.1
mutate_add_node_prob	0.0005	0.005	0.001	0.0025
mutate_add_link_prob	0.001	0.05	0.01	0.005
α	0	0	0.0000005	0.0000001
β	0	0	0.0000005	0.0000001
γ	0	0	0.000005	0.0000001
δ	0	0	0.000005	0.0000001
dropoff_age	100	50	100	200



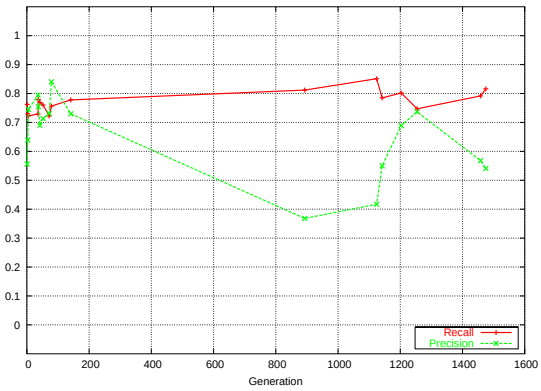
(a) *tr_seeded_inc5_A* - validation dataset



(b) *tr_seeded_inc5_B* - validation dataset



(c) *tr_seeded_inc5_C* - validation dataset

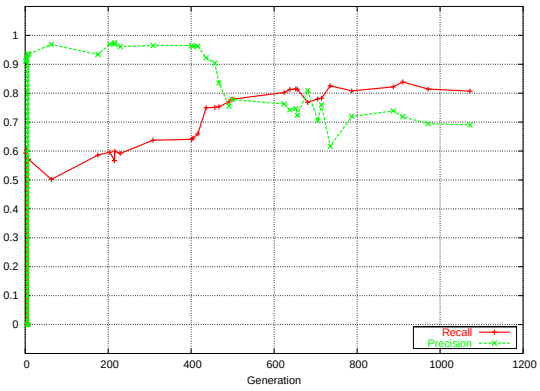


(d) *tr_seeded_inc5_D* - validation dataset

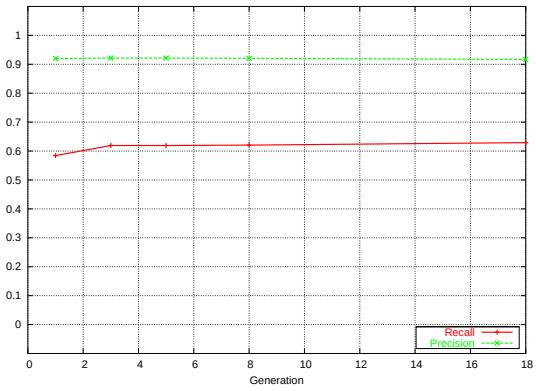
Figure 4.17: Seeded topology training run – iteration 5

Table 4.13: Parameters for iteration 6 of seeded starting topology training run

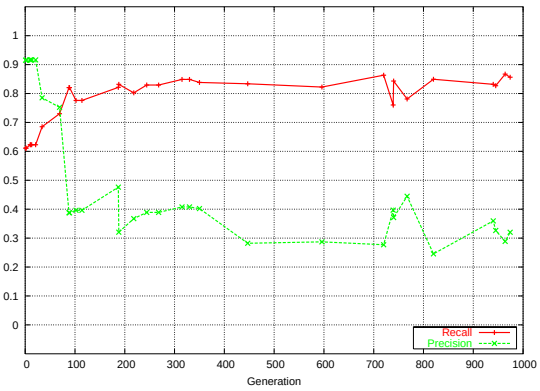
Parameter Value	Training Runs			
	tr_seeded_inc6_A	*tr_seeded_inc6_B*	*tr_seeded_inc6_C*	*tr_seeded_inc6_D*
Population	150	150	150	150
Total faces	1884	1884	1884	1884
Total random nonfaces	**7000**	**7000**	**7000**	**7000**
Total bootstrapped nonfaces	**13660**	**13660**	**13660**	**13660**
X_{face}	**400**	**400**	**400**	**400**
$Y_{nonface}$	**400**	**400**	**400**	**400**
false_positive_penalty	1.75	1.75	1.25	**2.0**
weigh_mut_power	0.01	0.5	0.25	0.1
mutate_add_node_prob	0.0005	0.005	0.001	0.0025
mutate_add_link_prob	0.001	0.05	0.01	0.005
α	0	0	**0.00000005**	**0.0000005**
β	0	0	**0.00000005**	**0.0000005**
γ	0	0	**0.00000001**	**0.0000005**
δ	0	0	**0.00000001**	**0.0000005**
dropoff_age	100	50	100	200



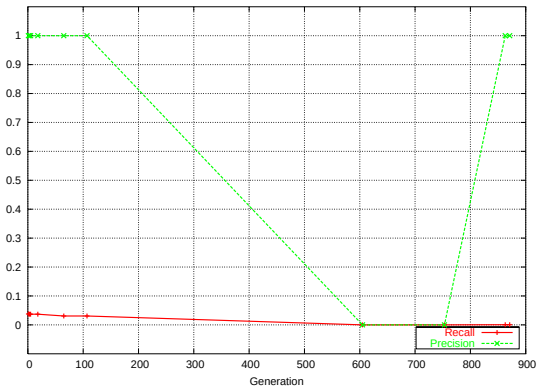
(a) *tr_seeded_inc6_A* - validation dataset



(b) *tr_seeded_inc6_B* - validation dataset



(c) *tr_seeded_inc6_C* - validation dataset



(d) *tr_seeded_inc6_D* - validation dataset

Figure 4.18: Seeded topology training run – iteration 6

4.5.1 Automatic Input/Feature Selection

The starting topology for the proposed training scheme (see Section 4) has a minimum of 292 inputs connected to the output. This means that there is already considerable structure (in terms of connection weights) for the GA to optimise. An alternative approach worthy of investigation is to connect far fewer input features (e.g. pixel intensity value) in the initial starting topology and rely on the "add link" mutation operator to make connections to the input features. This is attractive because it presents NEAT with a simpler starting point from which to evolve. Furthermore, as it is reasonable to imagine that perhaps not all inputs contribute to the information content of the face sample (particularly those pixels in the vicinity of the oval mask), evolving from a initial topology with unconnected inputs allows the GA to automatically select only input features which contribute to the final solution. This has the potential to further reduce the size of the topology. Unfortunately, using an initial topology with unconnected inputs was not possible using the version of the NEAT library which was used during the training phase of this research. However, the latest release of the NEAT software library allows for unconnected inputs in the starting topologies. For the reasons outlined above, this facility should be investigated in future work.

4.5.2 Using Alternative Low-Level Features

As was discussed in Chapter 3, a wide variety of low level input features have been investigated in prior appearance-based face detection research. The different low level features have varying characteristics, which have consequences for classifier training, rotation invariance, illumination invariance and computational cost. The proposed training algorithm uses the pixel intensity values directly as the input features to classify. An alternative approach worth investigating is to use frequency domain input features. Such features could be used in the proposed NEAT hardware accelerator without any further modification other than a frequency co-efficient normalisation step in software. Transforming pixel intensity values into the frequency domain preserves the original signal but redistributes the energy contained in the signal. For similar reasons to those motivating the use of frequency transformations in video compression (see Chapter 2), frequency based input features can be exploited to reduce the number of input features to NEAT. That is, high frequency coefficients, which will typically only contribute a small amount to the overall energy contained in the sample can be ignored. Alternatively, an incrementally increasing number of coefficients could be used in multiple classifiers in order to provide a course to fine grain cascaded classifier solution. A reduction in the number of inputs will greatly help to reduce the size of the evolved

topology. Whilst a smaller network is very attractive from a real time performance and low power perspective, arguably the greater benefit in the context of NEAT is that the GA has a considerably reduced search space to explore during training in order to find appropriate solution. This could lead to a more robust face detection solution. An additional benefit of using frequency-based input features is the potential for a very computational efficient means of sample illumination correction by simply ignoring the lower frequency co-efficients. For example, the DCT DC value and the next two coefficients (in zig-zag scan order) could be considered to represent the average and slowly changing horizontal and vertical gradients within a sample. Therefore, by ignoring these coefficients during the training phase, approximations to illuminations conditions such as shadows are also being ignored.

To ensure frequency-based input features were viable within the current hardware/software NEAT framework, preliminary investigations were carried out. The first design option encountered in these investigations was which of the many possible frequency transform based features to use. The options considered were wavelet coefficients (in particular the simplified Haar transform [113]), 20×20 DCT coefficients (with or without the oval mask) or 20×20 SA-DCT coefficients (see Chapter 2 for more details on the SA-DCT). In comparison to the simplified Haar transform values, DCT and SA-DCT coefficients were considered more straight forward to integrate quickly within the current framework. This was a considerable benefit owing to the nature of the preliminary evaluation. Although, it could be argued that using the DCT and SA-DCT are considerably more computationally demanding than the Haar transform. However, run-time was not a primary focus of these preliminary investigations. Furthermore, energy efficient hardware architectures for these transforms already exist [37]. The next decision was whether or not to use an oval mask on each sample. The disadvantage of the oval mask is that redundant information is being captured in the transform if pixels outside the mask are used to analyse frequency coefficients. This adds considerable energy to the higher frequency coefficients due to the sharp transition from black masking oval to the pixels of the face sample. Arguably the redundant masking oval information which is distributed throughout multiple coefficients makes a direct comparison between the frequency based approach with 400 maximum coefficient inputs and the pixel based approach with 292 input values unfair without further processing. The issue of fair comparison is exacerbated when not using the masking oval, as it potentially allows background structure to creep into the samples. Such background structure could also be detrimental to the training process. One possible solution to these issues is to use a SA-DCT where the oval mask acts as the alpha plane. The

SA-DCT will then only generate coefficients for the face pixels inside the oval mask.

A direct comparison between the intensity based training and frequency based training is possible when using 292 input pixels and 292 coefficients respectively. Although, completely different topologies should be expected, since the SA-DCT coefficients have a fundamentally different energy orientation than the pixel intensity values. Perhaps the more interesting question to explore though, is to examine if the detection performance degrades rapidly when using fewer coefficients or if in fact fewer coefficients can improve the detection performance due to a smaller search space for the GA to explore. Rather than select a somewhat arbitrary percentage of coefficients to investigate, a systematic approach was adopted, which tried to establish the average number of coefficients required to achieve a fixed level of sample energy. To achieve this, a 20×20 SA-DCT was carried out on ten randomly selected faces (see Fig. 4.19) from the face training dataset. Plots of the accumulated energy versus number of coefficients (zig-zag scan order) for each of the ten faces are shown in Fig. 4.20. From these plots it can be deduced that on average using 89 coefficients and 221 coefficients are sufficient to capture 95% and 99% of the energy respectively (see Table 4.14 for more details). Using this number of coefficients the ten random faces which underwent SA-DCT are reconstructed using the Shape Adaptive Inverse Discrete Cosine Transform (SA-IDCT). The reconstructed faces are shown in Fig. 4.19(b) and Fig. 4.19(c). The smoothing effect caused from a loss of high frequency coefficients is quite noticeable in Fig. 4.19(c), nevertheless each sample can be easily recognised as a face.

4.5.3 Detection of Side Profile and Rotated Faces

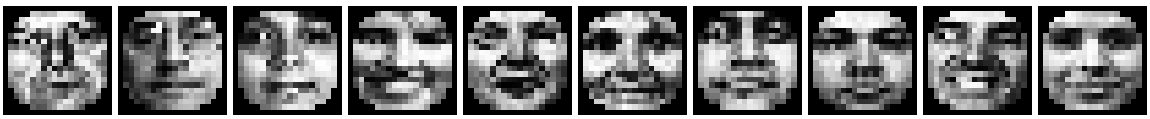
An obvious extension to improve the robustness of the face detection process is to include detection of in-plane rotated and side profile faces. An approach similar to that presented by Rowley could be easily used to achieve this goal [130].

4.6 Summary of Contributions

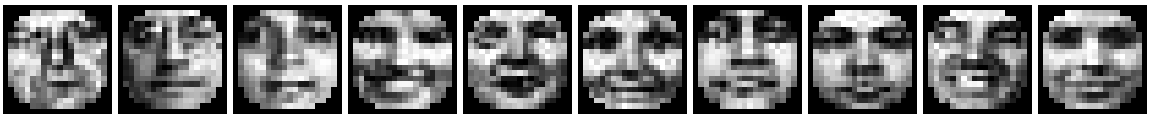
This chapter presented a novel EANN based face detection training algorithm. A detailed explanation was given in Section 4.2, which focused on the training data preparation, an overview of the NEAT based library in the context of the face detection training and finally the overall face detection training framework. The design decisions made in the training algorithm (e.g. stopping condition) were elaborated and discussed. Details of all the training runs were presented in Sec-

Table 4.14: Number of SA-DCT coefficients required to achieve specific energy targets

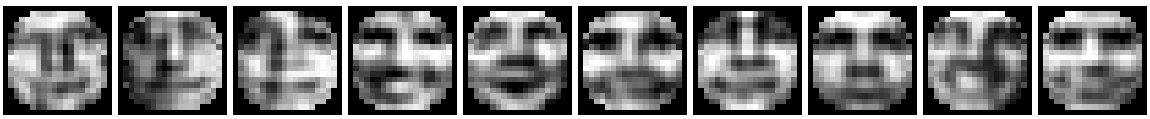
Image	**Number of coefficients required to reconstruct:**	
	95% of the energy	**99% of the energy**
Face 0	113	254
Face 1	88	209
Face 2	56	220
Face 3	75	206
Face 4	112	251
Face 5	103	233
Face 6	95	218
Face 7	64	203
Face 8	128	238
Face 9	56	174
Average	**89**	**221**



(a) 10 randomly selected faces from the face training dataset



(b) Faces reconstructed to the 99% energy level target using 221 coefficients



(c) Faces reconstructed to the 95% energy level target using 89 coefficients

Figure 4.19: DCT Energy exploration

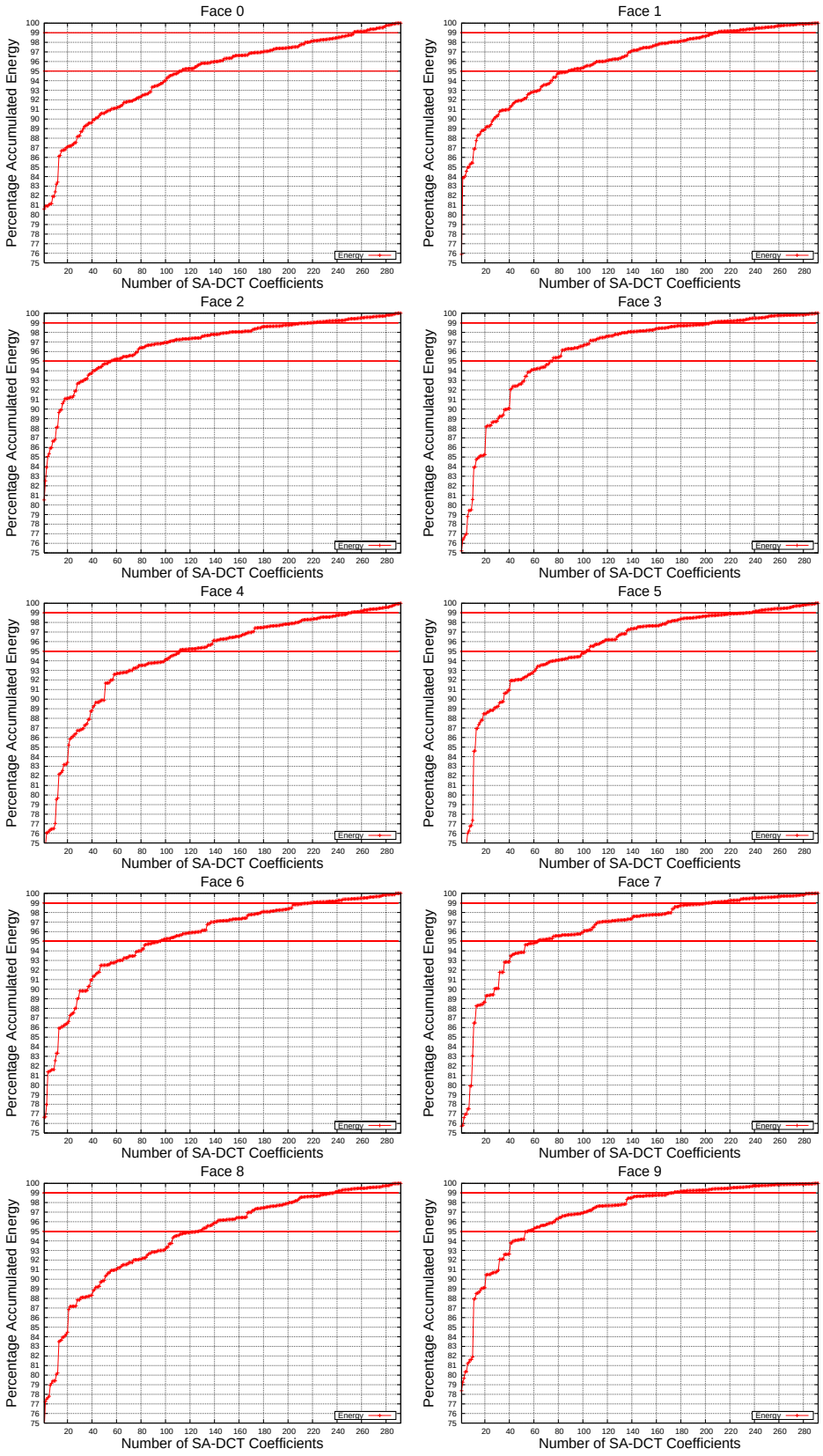


Figure 4.20: Energy versus the number of SA-DCT coefficients of each of the face samples

tion 4.3. An iterative approach to training was adopted which investigated two different starting topologies, a minimal non-seeded topology and a seeded starting topology. Parameter selection was extensively investigated during the training runs. It was found that the non-seeded training gave better generalisation performance with a smaller topology. It was concluded that the smaller initial search space of the non-seeded starting topology was the principal contributing factor for the improved performance. Using a large validation dataset the trained EANN achieved over 70% recall with 98% precision. The trained topology was compared to the generally accepted best performing ANN approach in the literature and was found to be considerably smaller [84]. This is advantageous for the intended target of a mobile device due to the reduced computational cost. Dynamic parameter selection in response to the progress of the evolutionary search was highlighted as a potential area of future research. This would also further automate the training process. Further avenues of future research include adopting alternative low level features and improving the robustness to rotation.

CHAPTER 5

Software Implementation of Trained Face Detection EANN

This chapter describes the normal mode operation (e.g. detecting faces during a video call) of the EANN based face detection algorithm which was trained in the previous chapter. When the training phase is complete, the best evolved solution is used in either software or hardware on the chosen platform. Section 5.1 describes the general algorithmic details of the normal mode operation. Profiling results are presented in Section 5.2, allowing design effort to be focused on computational complexity hotspots within the algorithm. Optimisations suitable for a software implementation are then described in Section 5.3. The profiling results also provide the motivation for the design of an energy efficient hardware accelerator. This hardware accelerator is described in detailed in Chapter 6. The chapter concludes with a thorough comparison of the proposed algorithm against other state of the art face detection algorithms using standard test datasets.

5.1 Normal Mode Face Detection Operation

To detect faces in an unseen image or video frame, overlapping 20×20 windows are extracted from the image. As with training, each candidate face window undergoes illumination correction, histogram equalisation and normalisation. The windows are selected from the image in a raster scan order, which from a hardware perspective results in an attractive regular addressing scheme. Similar to many other appearance-based face detection algorithms, in order to be able to detect faces larger than 20×20, the image is subsampled to produce an image pyramid (see Fig. 5.1)

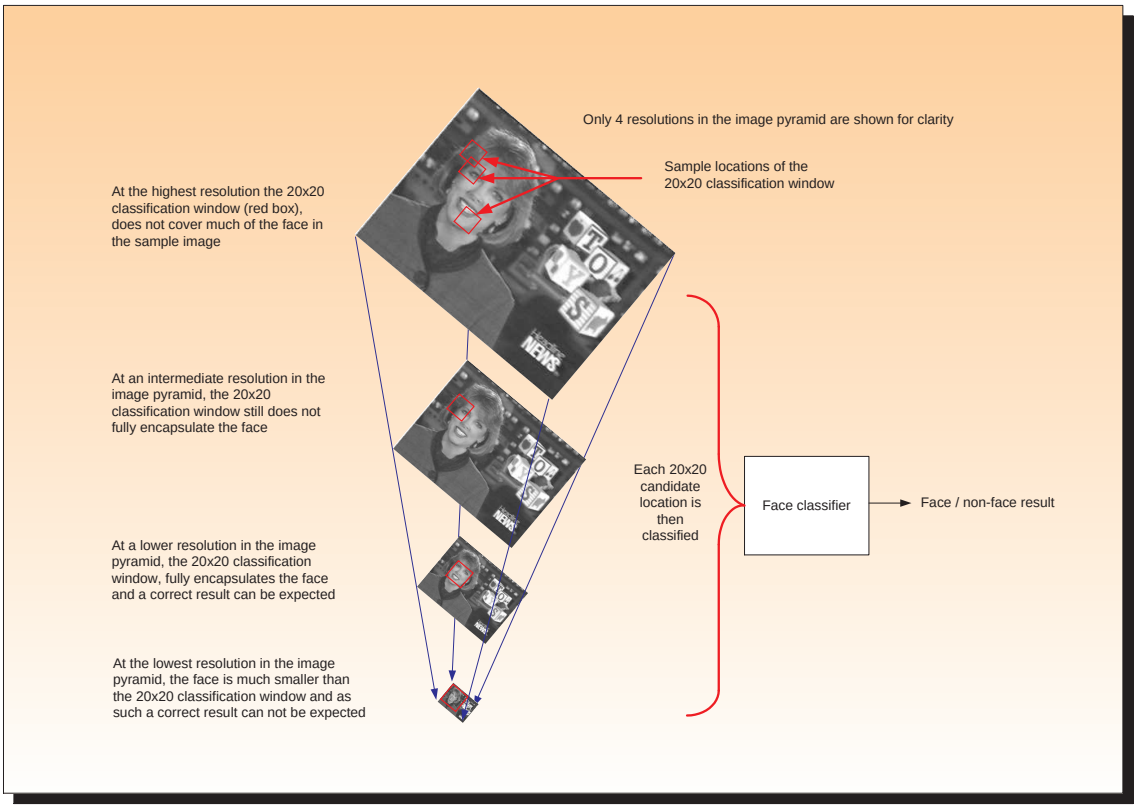


Figure 5.1: Example image pyramid

[84][116]. Bi-cubic subsampling with a reduction factor of 1.2 is used [84] [116]. On a computationally constrained platform, nearest-neighbour subsampling would also represent a viable alternative (see Section 5.3). Each 20×20 pixel window extracted from the image pyramid is subsequently applied to the evolved network.

Upon activation of the network, the output neuron gives a value between zero and one. The output is within these bounds due to the output range of the sigmoid activation function. The value of the output neuron can be considered to be the confidence level of the network that the candidate 20×20 pixel window contains a face. In order to improve the robustness of the final solution, rather than just using the single best evolved phenotype from the training, an ensemble of the fittest phenotypes is used instead. This improves performance because whilst multiple well trained networks tend to agree regarding correct classification, they agree much less frequently about incorrect classifications and this can be used to reduce the number of incorrect detections. This property is true provided there is some diversity between the distinct networks. For example, the diversity could be introduced by using different initial randomised weights or different topologies during training [84]. The NEAT library in the proposed approach provides inherent diversity in an evolved population through the presence of species. Therefore, the ensemble used in this work

consisted of the fittest member from the three fittest species. Different ways of combining the results from each phenotype to generate a final classification result were then investigated. These included Boolean combinations, averaging and a linear combination using precision estimates (generated from a validation dataset). In general, the linear combination approach gave the best performance. If the result of the combination of the phenotypes is above a specified threshold ($faceDet\_threshold$), the 20×20 pixel window is considered to contain a face. Reducing the threshold value increases the recall rate, however precision is then typically reduced due to the higher incidence of false positives. The choice of value for this threshold is discussed further in Section 5.4. The pseudo-code for the face detection routine is shown in Algorithm 4. Results from software profiling are discussed extensively in Section 5.3. In summary, the profiling reveals that the computational bottleneck (and by extension power consumption) within the algorithm is the repeated evaluation of the phenotype (shown as *species(i)→network→activate* in Algorithm 4).

5.1.1 Merging detection results

Even if the classification process always correctly identifies faces and non-faces, due to the overlapping detection windows the same face is likely to be detected at multiple locations. In addition, due to the image pyramid the same face is also likely to be detected at multiple resolutions. Therefore, before the final detection result for an image is available, all locations classified as a face must be examined to see if merging is possible. This property can also be exploited to reduce the number of false positives, as these are less likely to have the same occurrence pattern as correct detection results. For these reasons, face detection merging is commonly found in the appearance-based face detection literature. For example, Rowley et al, only considered a positive window result a face if the number of positive results within a local neighbourhood is above a threshold. Furthermore, if a face is detected across multiple resolutions, the overlapping faces at lower resolutions are eliminated. Similar heuristics were employed by Viola and Jones, Feraud et al, Garcia et al [113][116][83].

The approach taken in this work for merging face detection results, firstly compares the address (if necessary upsampled to the original image dimensions) and bounding box size (can be larger than 20×20 pixels if the detection is upsampled) against all other detections. It is then possible to establish whether there is any overlap in the bounding boxes of each detection. As the number of detections is relatively low in a single image (e.g. typical 5-20 detections for a single face in a QCIF image) this iterative step does not create an overly excessive computational burden.

Algorithm 4: Normal Face Detection Operation

```
load_train_population(input_population_file);
sp1_precision = estimate_generalisation(input_population→fittest_species1);
sp2_precision = estimate_generalisation(input_population→fittest_species2);
sp3_precision = estimate_generalisation(input_population→fittest_species3);
total = sp1_precision + sp2_precision + sp3_precision;
w1 = sp1_precision/total;
w2 = sp2_precision/total;
w3 = sp3_precision/total;
open file containing test files pathnames;
while (testfiles remained to be examined) do
    open test file;
    load input data from file;
    while (image subsampling still possible) do
        for (i=0; i<subsampled_image→height; i++) do
            for (j=0; j<subsampled_image→width; j++) do
                extract 20x20 pixel test_window;
                illumination_correction(test_window);
                histogram_equalisation(test_window);
                normalise_pixels(test_window);
                for (i=0; i<=3; i++) do
                    species(i)→network→load_sensors(test_window);
                    species(i)→network→activate();
                    classification(i) = species(i)→network→output_neuron;
                face_classification_result =
                classification(1)*w1+classification(2)*w2+classification(3)*w3;
                if face_classification_result>=faceDet_threshold then
                    upsample_address_of_detection_and_store;
                    store_confidence_of_detection(facedet) = face_classification_result;
                    faceDet++;
    merge_detections();
    write_out_resulting_image();
    testfile++;
```

If overlapping is detected, the challenge is then to merge the detections in terms of a new address, bounding box size and confidence level. A straight forward approach to this merging is to simply average the properties of the two detections. However, the author found that taking account of the relative confidence level and bounding box size of the two detections gave improved results for the bounding box size in the presence of non-ideal detections. The pseudocode for the proposed merging algorithm is shown in Algorithm 5. This scheme also helped to lessen the influence of false positives, which typically had a lower confidence level. To be considered a valid face detection result, the proposed merging algorithm requires the number of overlapping face detections to be above a specified threshold. This is a variant of the technique used by Rowley et al, and is used to reduce false positives [84]. The value of the threshold was derived experimentally.

Algorithm 5: Simplified pseudo-code for merging overlapping face detection results

```
for (i=0; i<=total_number_of_detections; i++) do
    for (j=0; j<=total_number_of_detections; j++) do
        if (overlapping(detection[i], detection[j]) && i!=j) then
            alpha_ratio = confidence[i]*(size_horz[i]/(size_horz[i]+size_horz[j]);
            beta_ratio = confidence[j]*(size_horz[j]/(size_horz[i]+size_horz[j]));
            new_block_vert_size = (int)((alpha_ratio*size_vert[i]) + (beta_ratio*size_vert[j])
            + 0.5);
            new_block_horz_size =
            (int)((alpha_ratio*size_horz[i])+(beta_ratio*size_horz[j])+0.5);
            new_top_left_vert = (top_left_vert[i] + top_left_vert[j])/2;
            new_top_left_horz = (top_left_horz[i] + top_left_horz[j])/2;
            new_confidence = (confidence[i]+confidence[j])/2;
```

5.2 Profiling of Normal Mode Face Detection Operation

This section attempts to establish the computational complexity hotspots in the proposed algorithm. Identifying these allows optimisation effort to be focused on the computationally critical aspects of the algorithm. The profiling is also useful in confirming the design decisions made regarding the suitability of off-loading the EANN evaluation to dedicated hardware. The algorithm was modelled using C++ and compiled using GCC 3.4 on an AMD-based server equipped with two dual-core 2.4 GHz processors (though only a single core was used). The profiling tool used was GNU gprof 2.15. In the interests of processor independence and fairness, each 20×20 pixel block was evaluated using a single phenotype rather than a ensemble of phenotypes. In this way, single and multi-core processors (which could evaluate multiple phenotypes in parallel) will generate similar profiling results. The profiling results are shown in Fig. 5.2, where it can be seen that the EANN activate function, EANN flush (i.e. reset the EANN) and load sensors (i.e. loading the input pixel values into the EANN) are approximately 95% of the total computational cost of the proposed algorithm. However, as highlighted during the algorithm design decision phase in Chapter 3, this functionality can be readily offloaded to dedicated hardware.

The time taken by the algorithm is principally a function of the processor architecture and processor clock frequency. To achieve independence from the processor clock frequency, the number of instructions and clock cycles required for the ARM-920T processor is quoted in Table 5.1. In the authors opinion, the ARM-920T is an appropriate choice as it is commonly found on mobile devices and is also representative of other 5-stage pipeline RISC processors found on mobile devices. The processing duration was recorded for the EANN classification functionality necessary for a single 20×20 pixel classification. This allows a fairer comparison later between comparable

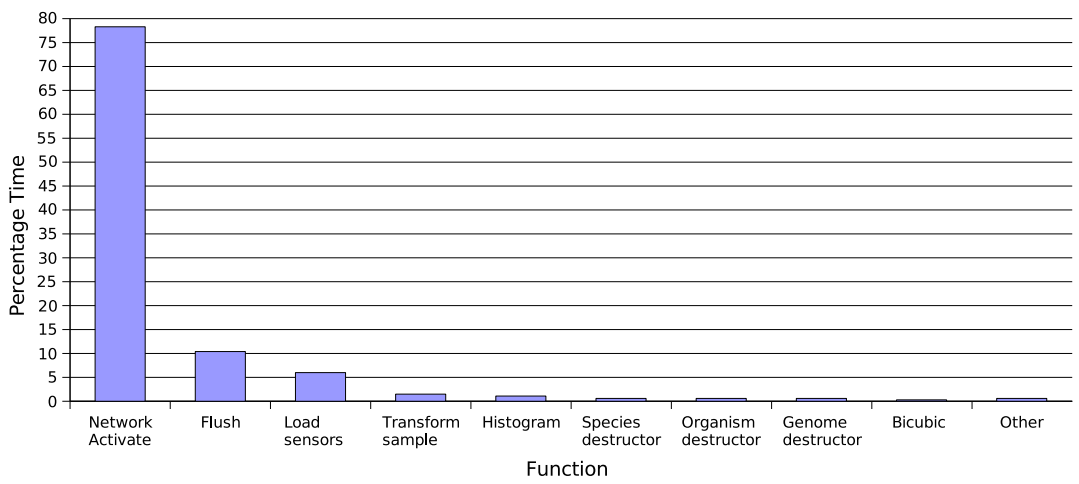


Figure 5.2: Software Profiling of Proposed Face Detection Algorithm

functionality in both software and hardware (see Section 6.4). As the EANN classification functionality was shown to consume 95% of the resources, this simplification can still be considered to give an accurate reflection of the processing duration of the algorithm on a mobile device.

The clock cycles for three different sized topologies are shown in Table 5.1. These topologies were extracted from the training phase (see Chapter 4). The processing required from 527,032 to 662,260 clock cycles to classify a 20×20 pixel block. It should be noted that the average clock cycles per instruction from the data in Table 5.1 is 1.57. This is relatively high for an ARM 920T (typical performance is 1.3 clock cycles per instruction [74]). This could be attributed to pipeline stalls originating from floating point calculations in the NEAT library due to the lack of a dedicated floating point processor. The increasing number of clock cycles can be attributed to the similarly increasing structure (i.e. hidden nodes and connections) present in the topologies. Even when using the smallest and (least accurate) topology, the evaluation time for a single 20×20 pixel evaluation when using a 120MHz clock frequency is $527,032 \times \frac{1}{120MHz} = 4.4ms$. This means only 227 locations can be examined in a second. To put this into context, examining all overlapping locations in a single QCIF image and ignoring lower layers of the image pyramid necessary for scale invariance still requires $(176-20) \times (144-20) = 19,344$ classifications. Clearly, a software only implementation of the proposed algorithm will struggle to achieve real-time performance on a mobile device. This is a common issue when attempting to implement appearance-based face detection algorithms on computationally constrained platforms. However, as already discussed earlier in this section, as well as in Chapter 3, the bottleneck constituting almost 95% of the total cost of the proposed algorithm is highly amenable for off-load to dedicated hardware. The design of such a hardware accelerator is the focus of Chapter 6.

Table 5.1: Clock Cycles Required to Classify a Single 20×20 Pixel Block

	Topology		
	A	B	C
Hidden Neurons	12	15	17
Connections	341	374	415
Instructions	332,639	374,871	418,655
Clock Cycles	527,032	593,914	662,260

5.2.1 Software Power & Energy Consumption

Having established that the ANN evaluation is the most computationally demanding aspect of the proposed algorithm, the power consumption of this functionality when implemented in software was calculated. This allows direct comparison of the energy efficiency of software and hardware implementations (this comparison is detailed in Section 6.4). The power consumed by a software program can be principally attributed to the logic switching in the host processor. In this research, the current drawn by an ARM920T processor was recorded, whilst it was running the software program, using the methodology proposed in [171][37]. In this way, instantaneous power can be easily calculated, assuming the voltage to the processor is fixed (e.g the development board used in this research used a fixed voltage of 2.5 volts). To generate meaningful power consumption figures, which allow a fair comparison between a hardware and software implementation, clearly the issue of non-optimised code should be addressed. The NEAT library makes heavy use of C++ standard template library functionality, which is only partially supported by the development tools for an OS-less implementation on the chosen platform. As a result, power profiling of the primitive functions (multiply-accumulate and the non-linear activation function) within an ANN was considered to give a fairer reflection of the software power consumption. Furthermore, as can be seen in Table 5.1, the more structure (i.e. neurons and connections) in the topology, the greater the number of clock cycles required to complete a single 20×20 pixel evaluation. Therefore measuring the software power consumption of the ANN primitives could be used to predict the overall power consumption of different topologies sizes.

The power consumed during ten evaluations of multiply accumulate operations is shown in Fig. 5.3. Similarly Fig. 5.4 shows the power consumption for a sigmoid activation function. In the case of both primitives, input data representative of that found in a NEAT ANN for face detection was used for the ten evaluations. The average power for the multiply accumulate operation is approximately 41mW, whilst 49mW was the average power consumed for the activation function. The higher power for the activation function is as expected, due to the more complex instructions

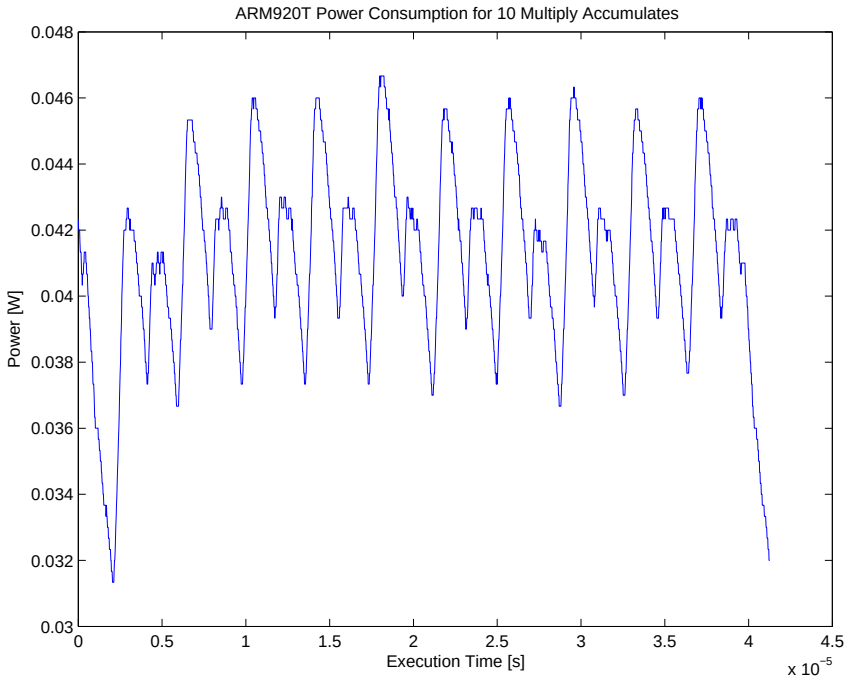


Figure 5.3: Power Profiling of Multiply Accumulate Operations on an ARM-920T processor

required for the e^x operation in the sigmoid function. Comparison with the power consumption of dedicated hardware is presented in Section 6.4.

5.3 Software Implementation & Algorithmic Optimisations

Whilst the majority of operations associated with the proposed (normal mode) face detection algorithms are arithmetic primitives (principally multiply accumulate in the ANN ensemble), the highly repetitive nature of examining the large number of candidate face locations in the image pyramid can greatly affect real-time performance. This is particularly true for computationally constrained mobile microprocessors. The issue is further exacerbated by using an ensemble of classifiers. Although this improves the reliability of the final solution, it also causes a linear increase (proportional to the ensemble size) in the run-time for a single processor. Profiling revealed over 90% of the computational resources required for the proposed algorithm is consumed by the ensemble of ANN classifiers. Fortunately, as was established in Chapter 1 and Chapter 2, a dedicated hardware accelerator offers a particularly attractive energy efficient solution for face detection on a mobile device, considerably reducing the significance of the outlined computational expense. As will be shown in Chapter 6, when using dedicated hardware it is possible to exploit the inherent parallelism in each individual ANN classifier to accelerate the classification process in an energy efficient manner. In addition, as there is no interdependencies between each ANN

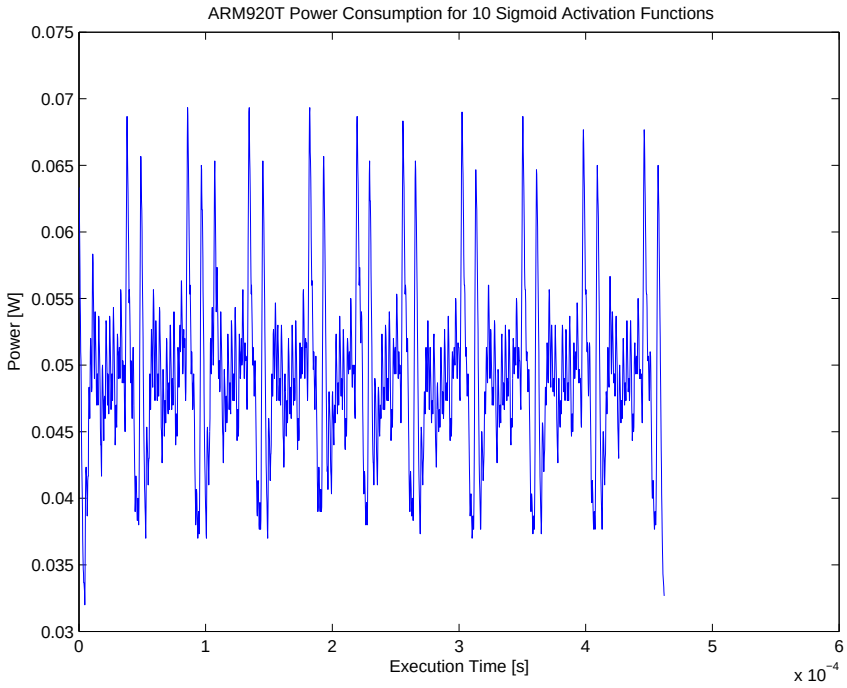


Figure 5.4: Power Profiling of a Sigmoid Activation Function on an ARM-920T processor

in the ensemble, a parallel implementation allows each ANN to operate concurrently further reducing the run time, albeit at a cost of increased silicon area. Furthermore, input pixels can be reused thus reducing accesses to memory, which is a considerable source of power consumption (see Chapter 2). Although a dedicated hardware accelerator is advocated, algorithmic flexibility is still preserved by implementing the non-speed critical elements and the control logic of the proposed face detection algorithm (i.e. image pyramid creation, extracting 20×20 pixel candidate windows, etc.) in software. This also increases the reusability of the ANN accelerator, since it is then not hard-wired to a single algorithm.

The dedicated hardware improves the power consumption required to classify a single location (see Chapter 6). To further improve energy efficiency it is necessary to reduce the number of classification locations. The cause of the high number of possible classifications is the requirement of the proposed algorithm (in common with all typical appearance-based face detection algorithms) to classify overlapping locations across multiple resolutions. This step is necessary for scale invariance, i.e. to be able to make correct classifications irregardless of the size of the face relative to the size of the image. However, the vast majority of overlapping classification locations will not contain a face. This is true even when considering an image with a large number of faces present (e.g. a crowd scene). Exploiting this property at an algorithmic level reduces computational expense, and by extension reduces the power consumption. The general theme in the literature to

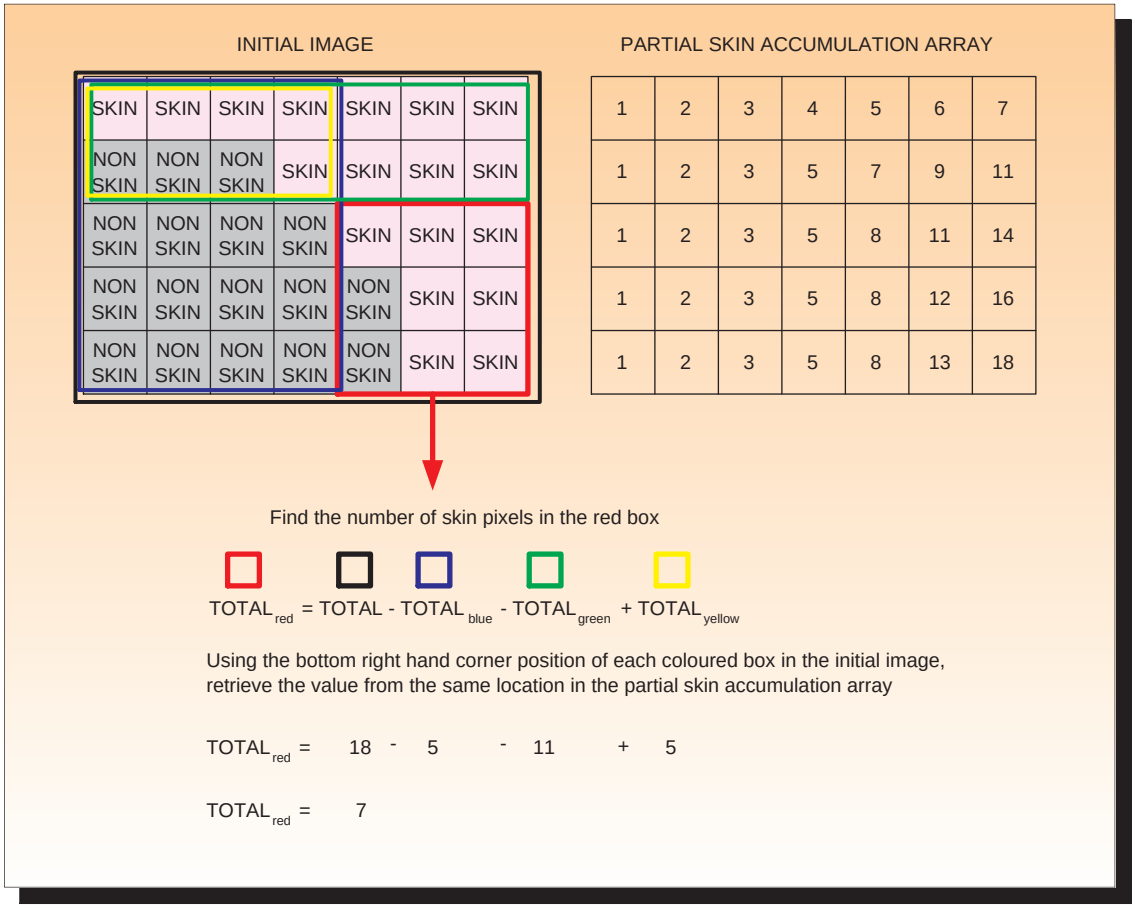


Figure 5.5: Efficient calculation of block skin totals

achieve this, is to eliminate unlikely classification locations quickly using as few computations as possible and to devote incrementally increasing computation time to regions where there is a higher likelihood of a positive result [113][116]. The optimisations to the proposed algorithm in terms of the modifications to the software control logic are now described.

As it is reasonable to expect colour source images (e.g. from a mobile phone camera) it is possible to use skin detection as a preliminary filtering stage. This is similar to the approach taken by Feraud [116]. The skin detection algorithm used is the computationally efficient chrominance thresholding scheme described in Section 3.1.1.1. If, after the skin detection process, the resulting total detected skin pixels for the block is below a predetermined threshold, no further processing is carried out and processing moves to the next location. As well as a direct implementation of the skin detection pre-processing, an optimised implementation was investigated which used a modified version of the integral image scheme used by Viola and Jones adapted for skin detection [113]. In this implementation rather than calculate the skin pixel total for all overlapping blocks at all resolutions, an intermediate array is used to store the sum of the skin pixels to the left

and above each pixel (inclusive of that pixel) at the highest resolution (i.e. original image). The advantage of generating the partial accumulation array is that the number of skin pixels in any subsequent block (at any resolution) can be calculated with one addition and two subtractions (see explanation in Fig. 5.5). This compares favourably to the 100 threshold operations and up to 100 accumulations (assuming subsampled 10×10 chroma blocks) for a direct implementation. Therefore, with the optimised implementation, following a comparison with the skin threshold value, a candidate location could be potentially skipped in 4 operations. The skin detection pre-processing was further extended and allowed to dynamically alter the horizontal address step size in proportion to the number of skin pixels found at the current location. For example, if the skin pixel threshold for a 10×10 chrominance block was 50, and no skin pixels were found at the current location, it is reasonable to advance the chrominance horizontal address by five rather than just one. In the ideal scenario this reduces the computational burden associated with processing overlapping locations. An alternative approach to reduce the burden of overlapping locations is the method used by Rowley et al, whereby a 30×30 window is used for classification, but which was trained with off-centred (± 5) 20×20 pixel face samples. The 30×30 window is then advanced 10 pixels at a time. A finer grade search using a more accurate 20×20 pixel classifier then inspects only the more promising locations from the course grain search.

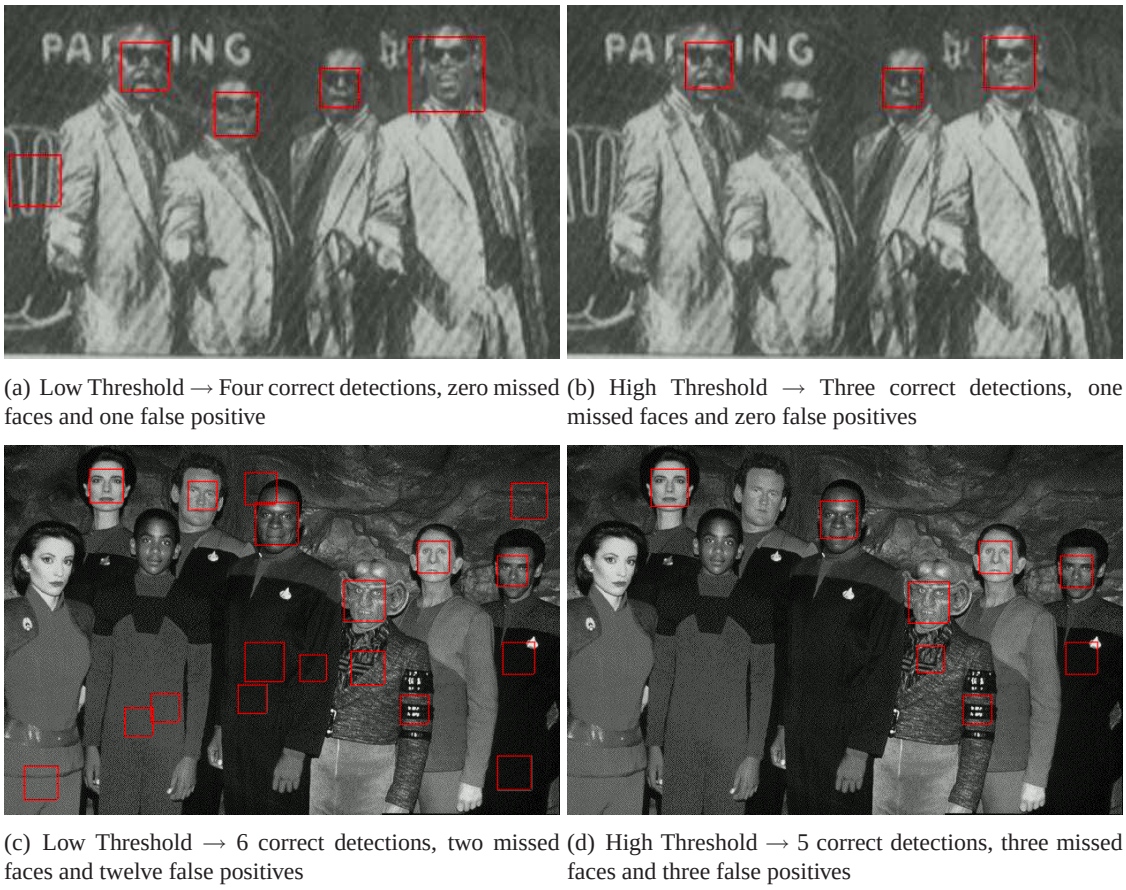
The smallest sized face that can be detected by the algorithm is 20×20 pixels. This allows very small faces to be detected and up to approximately 238 faces to be detected in a CIF-sized image (assuming non-overlapping faces). However, detecting faces this small or detecting this many faces would be atypical of the type of applications which could benefit from face detection on a mobile device. Therefore a reasonable restriction is to enforce a larger minimum detectable face size. To achieve this, rather than increase the classification window size, it is preferable to skip resolutions where the upsampled 20×20 classification window size is smaller than the minimum detectable face size. This is attractive from a computational perspective as it greatly reduces the the number of locations to classify.

To further reduce power consumption, the subsampling mechanism is altered from bicubic subsampling to use the nearest neighbour subsampling technique. Not only does this result in fewer computations, nearest neighbouring subsampling does not require a local context of pixels for the generation of the subsampled pixels. This leads to a reduction in highly power consumptive memory accesses. Finally, functions that were deemed suitable for further optimisation, which was established from profiling, were targeted with low level optimisation techniques appropriate for a

mobile processor such as an ARM. These techniques included using loop unrolling, using appropriate data types for an ARM processor (e.g. *INT* instead of *SHORT INT*), where possible avoiding the use of floating point arithmetic, etc. For example floating point operators were replaced with integer arithmetic approximation in functions such as colour space conversion (YUV→RGB, RGB→YUV). These combined optimisations allow a software only version of the algorithm to classify a QCIF-sized image in less than 10 seconds on a 120 MHz ARM 920T microprocessor which was concurrently running Linux [172]. Detection results and performance throughput are discussed further in Section 5.4. Other possible optimisations for reducing power are left as future work, but are discussed in Section 5.5.

5.4 Results

To verify correct operation and to assess the quality of the proposed solution, this section benchmarks the proposed face detection system against prior related research. The proposed algorithm is compared against state of the art face detection approaches in software, thus allowing evaluation of the algorithmic performance in terms of recall accuracy and precision. Although, the FERET database was used in the generation of the positive face training dataset, it is not appropriate for face detection benchmarking, as this could give a biased reflection (i.e. excessively positive) of the detection performance due to using evaluation samples very similar to the training dataset. Instead, alternative datasets which are also representative of the "real world" problem of face detection on a mobile device are used. Examples of such include: the CMU/MIT face database, the Yale face database and the BioID face database [84][115][165][166]. The Yale dataset is most frequently used for face recognition benchmarking and as such is excluded from the benchmarking process of this section. The CMU/MIT dataset is often used for face detection benchmarking, however in the author's opinion, the low resolution of the majority of test images (many having a "newspaper" photograph quality) make it unrepresentative of the medium to high resolution images/videos generated from current mobile devices equipped with cameras. For this reason the BioID dataset is used as the principal means of evaluating the performance of the proposed face detection algorithm. It should also be noted that a further benchmarking complication arises when prior related research efforts have evaluated their algorithms using custom testing datasets. As well as this making fair benchmarking very challenging, there is a risk of testing on the training set and/or tweaking the algorithm for the custom evaluation dataset. This can lead to a performance



(a) Low Threshold → Four correct detections, zero missed faces and one false positive

(b) High Threshold → Three correct detections, one missed faces and zero false positives

(c) Low Threshold → 6 correct detections, two missed faces and twelve false positives

(d) High Threshold → 5 correct detections, three missed faces and three false positives

Figure 5.6: Effect of changing the face detection threshold (images taken from the CMU/MIT dataset)

distortion, masking the algorithms ability for "real world" generalised face detection.

A further tool used in the face detection performance evaluation is Receiver Operator Characteristics (ROC) graphs. These are commonly used in the evaluation of face detection algorithms, as the performance can be made to vary with operational parameters. For example, in the proposed algorithm by reducing the face threshold value (i.e. the EANN result for a 20×20 pixel block must be above the face threshold value to be classified as a face) more faces will be identified. However, an increasing number of non-faces will also be incorrectly classified as a face. This characteristic is demonstrated in Fig. 5.6. In addition, this property can also affect the performance of the merging algorithm. In many cases, fewer detections in the vicinity of a face will result in a more correctly centred bounding box. In contrast, a lower threshold will probably have more correct detections in the neighbourhood of a face, but may also have incorrect detections close by, which could distort the final result (incorrect bounding box size or off-centred bounding box) from the merging algorithm. The optimal trade-off point is application dependant, where a cost can be associated with a false positive.

Table 5.2: Benchmarking results for proposed algorithm against the BioID face database

Algorithm	**BioID Evaluation Dataset**	
	Recall (%)	False Detections
Ramirez & Fuentes	95.13	5240
Haung et al	90.00	n/a
Froba & Kublbeck	95.67	64
Froba & Ernst	97.75	25
Wu & Zhou	94.50	n/a
Jesorsky et al	91.80	n/a

5.4.1 Evaluation on the BioID face database

A selection of representative results generated using the proposed algorithm with the BioID face dataset are shown in Fig. 5.7. A recall rate of 93% with 12 false positives from 100 images[1] was recorded (see Fig. 5.8 for ROC graph). Interestingly, increasing the face detection threshold reduces the number of false detection as expected, however somewhat unexpectedly the number of correct detections also increases. This could be explained by the fact that false positives in the vicinity of correct detections sometimes reduce the performance of the merging algorithm, in that on occasions the merged result is outside the boundaries of the face. For example, if numerous false positives that could be deemed as weak true positives (containing only a small fraction of a face) happened to overlap a single true positive, the merging algorithm could easily give a final detection result that did not include the face. Whereas, with fewer false positives surrounding a correct detection the merging algorithm appears to perform better.

One disadvantage of using the BioID face dataset is that there is less prior research to compare against. For example, the AdaBoost based face detection algorithm by Viola and Jones, which is considered one of the leading approaches of recent times (see Section 3.2 for a discussion of the proposed algorithm compared to this approach), does not use the BioID dataset. Therefore, for completeness the author used the OpenCV/Blepo implementation of the Adaboost algorithm to generate BioID results appropriate for comparison [173]. A secondary benefit of doing this, was that it gave a controlled testing environment to compare the face detection results. It was found that Viola and Jones' algorithm detected 95% of the faces with no false positives. The proposed algorithm could be considered to give comparable performance (i.e. 93% detection with 12 false positives) to Viola and Jones' algorithm. Moreover, as will be discussed shortly, the proposed scheme is also competitive with other face detection algorithms that used the BioID dataset (see

[1]Only the first 100 images from a total 1200+ images in the BioID dataset were used due to performance run-time issues

Figure 5.7: Representative results from the BioID face evaluation dataset

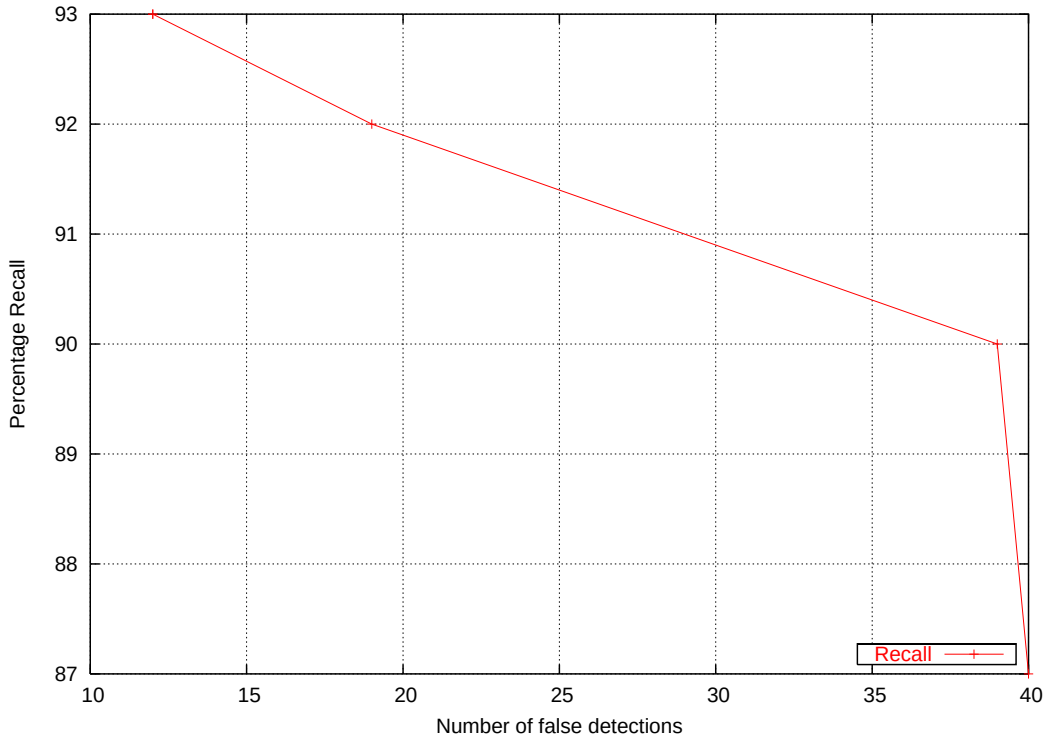


Figure 5.8: ROC graph generated from proposed algorithm when using the BioID dataset

Table 5.2).

Whilst the BioID dataset is used less frequently compared to some other face datasets, there is still a substantial number of research publications in which it is used. A brief overview of a representative sample of these publications will now be given. Ramirez and Fuentes used a two stage approach, in which the first stage uses rules regarding the position and contrast of the eye regions along with analysis of histograms [174]. The first stage yields a binary results of promising candidates, which is then used in second stage with a combination of classifiers including a SVM, voted perceptrons, an ANN and a naive Bayes classifier. A detection rate of 95.13% is achieved, though false positives are in the order of several thousand. Huang et al uses a radial symmetry based transform coupled with eye position checking [175]. This approach achieves a detection rate of 90%. Froba and Kublbeck uses edge orientation features to detect faces followed by a SNoW classifier for verification [176]. This achieves a detection rate of almost 96% with a low number of false detections. An alternative approach by Froba and Ernst that uses a census transform achieves a higher detection rate of 97% with a lower false positive detection rate. Wu and Zhou use a selective attentional mechanism [177]. This feature based approach gives a detection rate of 94.5%. A hausdorff distance based approach by Jesorsky resulted in a an almost 92% detection rate [166]. Overall, it is clear that the proposed algorithm is competitive with prior research when

benchmarking against the BioID face dataset.

5.4.2 Failure Analysis on the CMU/MIT face database

A comparison of the performance of the proposed algorithm against typical state of the art face detection algorithms using the CMU dataset is shown in Table 5.3 with representative results shown in Fig. 5.9. Unfortunately, it can be concluded that the current revision of the proposed algorithm does not perform as well as the leading prior research in the area. The recall rate of approximately 50% is considerably lower than prior related research.

In the author's opinion, this could be attributed to a number of factors. One of the principal motivations for the proposed algorithm (as outlined in Section 3.2) was the scope to minimise the computational complexity through the novel evolutionary training algorithm. Reduced computational complexity could be considered vital for mobile device operation, but it does appear that there is some trade-off in the general face detection performance associated with the proposed algorithm. However, it could be argued that increased battery life compared with reduced face detection performance is an acceptable trade-off on a mobile device. Furthermore, it should also be noted that the CMU dataset is not typical of the type of content generated from current generation mobile devices. Many of the test images have low resolution and difficult environmental conditions (i.e. illumination variance).

Reduction in the performance of the algorithm could be attributed to an insufficient number of training images of faces under different rotations. In addition, the results highlighted a need for a revised exploration of the modelling of illumination variance, as well as a need for a more robust merging algorithm. In particular, the merging algorithm on occasions (e.g. crowd scenes with faces located close together) can be overly aggressive. Also worth noting is that as expected when the face threshold increases the number of detections decreases, but there is also a dramatic reduction in the number of false positives (see Fig. 5.10).

5.5 Future Work

The deployment of the trained EANN topologies can be improved in terms of speed (and power consumption), detection performance (i.e. recall & precision) and increased algorithmic robustness (i.e. allowing detection with different face orientations). These broad categories of optimisations should focus on the distinct logical boundaries within the algorithm, namely the input stage,

(a) (b) (c)

(d) (e) (f)

(g) (h) (i)

(j)

Figure 5.9: Representative results from the CMU face evaluation dataset

Table 5.3: Benchmarking results for proposed algorithm against the BioID face database

Algorithm	**CMU Evaluation Dataset**	
	Recall	False Detections
Rowley et al [84]	86.2%	23
	84.5%	10
	77.0%	8
Feraud et al [116]	86.0%	8
	84.0%	4
	83.0%	3
	81.0%	1
Viola and Jones [113]	88.4%	31
	76.1%	10
Li and Zhang [111]	90.2%	31
	83.6%	10
Garcia et al [83]	93.3%	197
	90.3%	8
	88.8%	0
Proposed Solution	49.4%	240
	46.0%	167
	46.0%	159
	42.3%	160

the NEAT classifier and the search strategy used during normal mode operation. In the input stage, the use of the integral image technique proposed by Viola and Jones would be worth investigating [113]. Compared to the image pyramid technique, the integral image may lead to improved speed. As highlighted in Section 4.5.2, alternative frequency based input features should also be explored so as to potentially reduce the number of EANN inputs and consequently the overall size of the trained topologies. Frequency based input features would also be inherently suitable for a cascade of increasingly complex and more robust classifier stages. For example, a small number of lower order DCT coefficients could be used in a very small and very fast topology. Such a classifier is likely to have a very high false positive rate, therefore increasingly computationally complex classifiers (using an associated increasing number of DCT coefficients) would be required to refine the result. The overall benefit of such an approach is that computational resources are directed to promising face candidate locations, whilst obvious non-face candidate locations are rejected quickly.

As highlighted in Section 5.4, the performance of the algorithm is a critical area which needs to be improved. This could be achieved through additional training iterations using an increased number of training samples (e.g. considerably more rotated training samples and samples with the addition of random noise). Further bootstrapped training data could also be used. It is reasonable

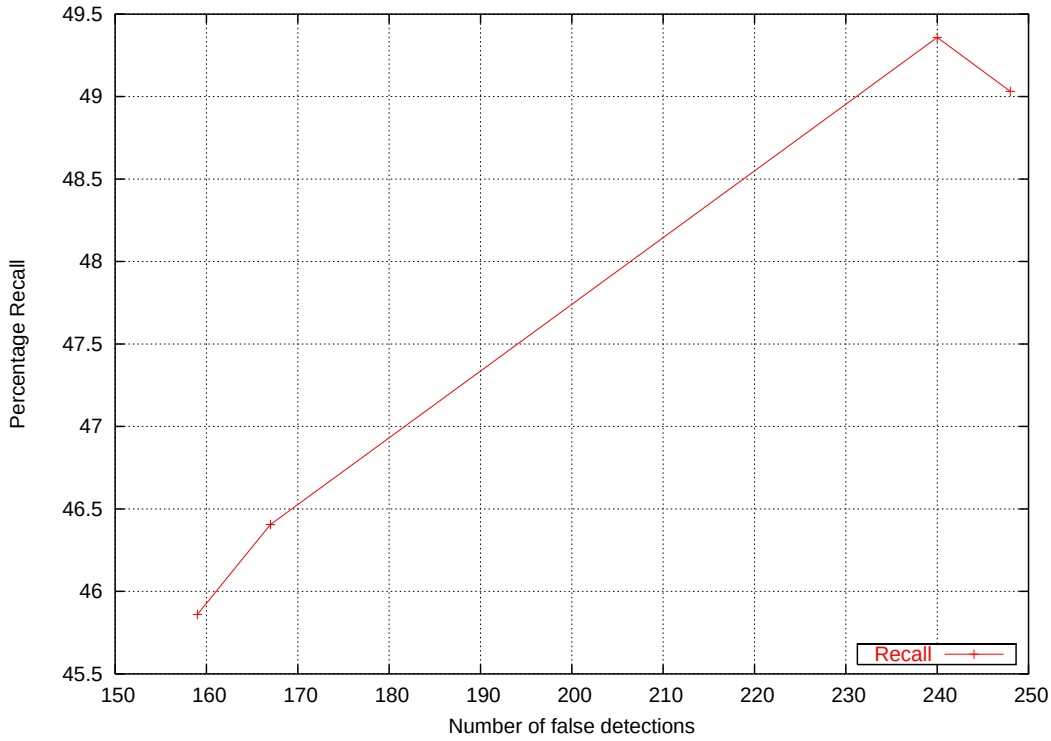


Figure 5.10: ROC graph generated from proposed algorithm when using the CMU dataset

to expect such additional training iterations would lead to improvements in both the recall and precision. It should also be noted that as the current ANN topologies are 50% to 75% smaller than that used by Rowley et al there is considerable scope for additional training to generate more robust solutions which could still offer computational complexity benefits. Another obvious extension to improve the robustness of the algorithm is to include detection of in-plane rotated and side profile faces. An approach similar to that presented by Rowley et al could be used to achieve this goal [130].

A further interesting avenue of research is to apply an “active vision” type face detection mode. In such a mode, rather than raster scan every pixel block location at multiple resolutions, as is typical with appearance-based face detection algorithms, the large search space could be explored using GA principles. Each candidate in the population would consist of a candidate pixel window location address, a scale/zooming factor (for face scale invariance purposes) and a rotation factor. The genome fitness would be a combination of the EANN response and/or factors such as skin count. As such a GA driven scheme is likely to discover inherent properties of a particular system, a reduction in the overall number of locations to inspect is quite likely. This would greatly benefit throughput and power consumption. As an example of the type of properties the GA could exploit, consider a driver of a car who typically scans locations in their visual field, however not

all locations have the same level of importance. Similarly in a mobile device video conferencing application, the face is likely to centred and within a relatively narrow range of scales.

Finally for optimum performance on a computationally constrained mobile platform, the NEAT library should be rewritten with such a target platform in mind (however this would require substantial effort).

5.6 Summary of Contributions

This chapter firstly presented a thorough description of the functioning of the proposed face detection algorithm during normal operation (e.g. detecting faces during a video call). This description included a discussion of the design choices made in selecting the classifier ensemble and face detection merging. Profiling was then carried out to identify the most computational complex elements of the algorithm. This identified the EANN evaluation and the associated tasks of loading the inputs and reseting the topology) as requiring almost 95% of the total computational load of the algorithm. This computational load characteristic is as expected, since the face detection algorithm was selected based upon potential to offload the most computational demanding aspect to dedicated hardware (see Chapter 3). In addition to function-based profiling, run-time evaluation and power profiling results were presented for an ARM-920T microprocessor. This identified that a software implementation of the proposed algorithm on a typical mobile processor would struggle for real-time operation. This provides strong motivation for the design of a dedicated hardware EANN accelerator, which is the focus of the subsequent chapter. In addition to the dedicated hardware, the other complementary method of reducing the computational complexity of the algorithm is to reduce the number of candidate face locations classified using the EANN. This was addressed by using skin detection pre-filtering of each candidate face location. Further software optimisation appropriate for a mobile device software implementation were discussed in Section 5.3. The performance of the proposed algorithm was then compared against state of the art face detection algorithms. The algorithm performed competitively with related research on the BioID dataset which could be considered representative of the image quality and facial pose found in the target mobile device application. Furthermore, a number of potential ways of improving the performance were highlighted and identified as future avenues of research.

CHAPTER 6

EANN Hardware Accelerator

It is vital for mobile computing platforms that fundamental enabling technologies, such as an ANN, can be deployed in a manner to meet the strict requirements of energy efficiency and real-time performance. Unfortunately, when processing high dimensional datasets, for example digital video, an ANN may not meet system requirements. In the case of digital video, poor performance is likely to be a consequence of the unavoidable combination of a requirement for extremely high throughput for real time processing and large complex ANN topologies. As the review in Chapter 3 showed, typical ANN based face detection systems use topologies with hundreds of inputs, thousands of connections and often as many as 100 hidden neurons [84][116][83]. This in itself would not pose a problem if only a single classification result was required, however the ANN processing becomes a major computational expense when it is repeated for overlapping pixels blocks across multiple resolutions as is commonly the case for face detection. The computational expense is further exacerbated by requirements for an ensemble of ANN classifiers. The associated computational complexity is highly undesirable on constrained computing platforms (i.e. mobile devices) from both real time operation and low power consumption perspectives.

As discussed in Chapter 3 one method of reducing the complexity is to use a minimal topology. This is beneficial since fewer neurons and connections lead to reduced computation, which in turn means higher throughput and lower power consumption. However, finding even a non-minimal network topology and appropriate set of weights represents a difficult multidimensional optimisation problem. Ideally, the topology should be as small as possible, but large enough to accurately

fit the training data. Failure to find a suitable configuration will cause poor generalisation and/or excessive execution time. As Chapter 3 highlighted one possible solution to this issue is to use an EANN (see Section 3.3.3 for more details), in which a genetic algorithm evolves an optimum topology and/or weights. As well as reducing the requirement for trial and error design exploration for the ANN, the approach is better equipped to avoid local minima and has scope for finding a minimal topology [131].

However, even with minimal topologies, when carrying out EANN pattern classification (after EANN training) on a mobile device, it still may not be capable of achieving real-time performance. This was demonstrated by profiling in Chapter 5, where even after considerable software optimisations, the proposed EANN-based face detection algorithm still required in excess of 10 seconds to process a single QCIF sized frame on a 120MHz ARM 920T processor. Whilst it may be reasonable to expect a higher clock frequency on mobile devices of the future, real-time software based face detection in video sequences with such an algorithm will remain extremely challenging. Moreover, whilst in the future a faster processor may to some extent tackle the real-time performance issue, the problem of energy efficiency due to the computational expense will remain. As alluded to during the description of the proposed face detection algorithm in Chapter 5, one possible solution to these issues, is to offload the computational burden from the host processor to a dedicated hardware accelerator.

The design of such a hardware accelerator is the focus of this chapter. The goal of this hardware core is to accelerate the face detection algorithm which was described in Chapter 5, in an energy efficient manner. Furthermore, this core should ideally be flexible enough to accommodate other ANN tasks appropriate to a mobile device. To achieve this flexibility a hardware / software ANN co-design methodology is adopted. As was also observed by Reyneri, this approach to ANN hardware design combines the scalability and flexibility of software with the speed and energy efficiency of dedicated hardware [155]. For example, by modifying the software the same core that accelerates face detection could be reused as an accelerator for advanced AI for computer gaming [31]. A block diagram of an ARM microprocessor (commonly found on power constrained platforms) based hardware integration framework suitable for this example is shown in Fig. 6.1 (repeated here from Chapter 1 for the sake of clarity). In the face detection scenario, EANN based training would occur once. After training, during normal mode face detection operation (e.g. face detection during a mobile video call), the proposed algorithm will leverage the accelerated processing performance of the dedicated core. In the AI application scenario, constant GA based

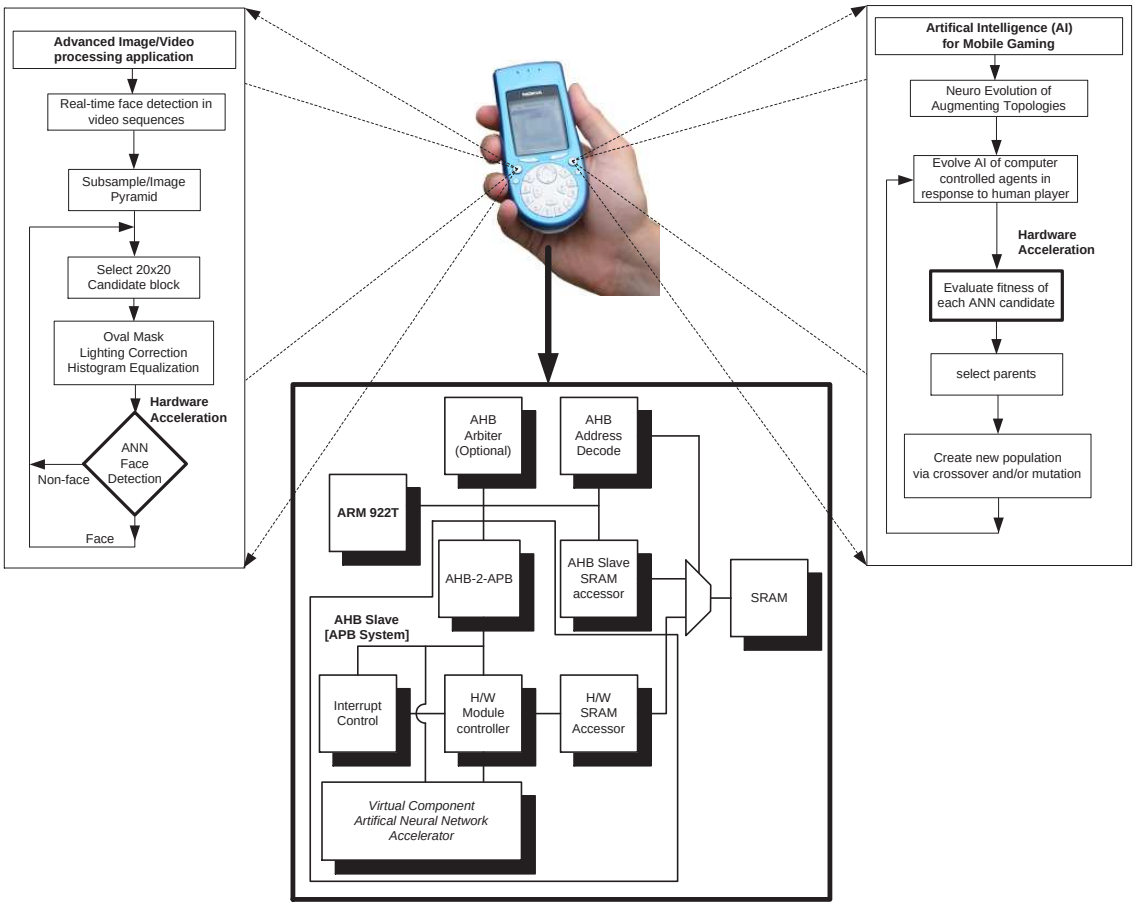


Figure 6.1: NEAT multimedia hardware accelerator

evolution with hardware accelerated phenotype evaluation would occur to alter the AI of computer controlled agents in response to the playing characteristics of the human player [31]. As was discussed in Chapter 3, the computational burden of the ANN evaluation is suitable for hardware offload, whilst the GA, which uses moderate computational resources resides in software. However, to fully exploit the possibilities offered by the EANN paradigm, the unique features of EANNs have a considerable impact on the hardware implementation of the ANN compared to a traditional ANN implementation. Certainly the EANN approach is attractive from a hardware perspective relative to an ANN trained via on-chip back propagation as the GA optimisation search is less sensitive to reduced word length implementation issues. This and other implementation issues are described further in Section 6.2.

This chapter will firstly present a detailed review of ANN and EANN hardware architectures in Section 6.1. As hardware implementations of EANNs are limited, the focus of the hardware review will instead be on ANNs, under the assumption that EANN training can take place in-the-loop. A detailed description of the proposed hardware architecture is given in Section 6.2.

A comparison against prior related research in terms of synthesis results and power consumption where possible is carried out in Section 6.4.

6.1 State of the Art Hardware EANN Review

There is a considerable body of research in analogue, digital and hybrid hardware ANN implementations [178][179][180][181][182]. Analogue implementations typically have the benefit of high speed operation, smaller silicon footprint and lower power consumption compared to a digital implementation. However an analogue design has a number of drawbacks including susceptibility to component process variation, electrical noise and environmental conditions, along with resolution issues for the weights and activation function. A digital hardware implementation, on the other hand, does not suffer from these drawbacks, and furthermore allows ease of design and computational accuracy. For these reasons a digital design approach is adopted in this research.

A digital ANN implementation typically stores the network topology and/or weights in memory. These values are then later retrieved and processed in discrete chunks by the parallel PE. This is advantageous because any network size and potentially any topology can be handled. A PE usually consists of multiply accumulate circuitry [183][184][185] and sometimes depending on the architecture, an activation function generator. The number of processing elements implemented is a design space trade off between area, power and performance. This design space extends from a single PE to a PE for each neuron or even a PE for each connection calculation. A single PE could be compared to a conventional ANN software program operating on a single desktop or mobile processor. However, such an implementation is likely to suffer from throughput issues with high dimensional ANN data tasks. On the other hand, using one PE per neuron or connection will give excellent throughput but at the cost of prohibitive area and power consumption. A discussion of PE implementation challenges is given in Section 6.2.

One of the principal challenges for ANN acceleration using time shared PEs is how to get the ANN input data from slow, bandwidth limited memories to the high speed PEs in a fast, bandwidth efficient manner, which maximally exploits the throughout of the PEs. Systolic array architectures have been the pervasive choice for bridging this gap [186][187][188][189][184][190][191][192][193][185]. It is well established that systolic arrays offer many benefits for ANNs with regard to using memory bandwidth effectively by maximising data reuse, in addition to permitting highly regular PE control logic [194]. However, as will be discussed in Section 6.2, topologies with

sparse connections considerably reduce the efficiency of systolic array approaches, in addition to increasing memory requirements. Sparse topologies are attractive from a low power consumption perspective as obviously fewer connection computations are required. Despite this, the literature contains very few hardware implementations, which attempt to exploit topology sparseness [195]. The literature rather focuses on dense connectivity neural algorithms [196].

6.1.1 Efficient Multiply Accumulation

In a system where the ANN weights are static, canonic signed digit representation and multiplier-less distributed arithmetic architectures can be employed in the generation of the weighted sum [197]. However, static weights are clearly not suitable for EANN-based training. Even if EANN training takes place offline, such an implementation for the resultant fittest phenotype would restrict the reusability greatly, since the weights are clearly optimised for a single application. Fortunately, owing to the prevalence of the sum of products operation in digital signal processing algorithms and matrix arithmetic, there is a considerable research on efficient hardware implementations of this arithmetic primitive [198][199][200][201][202]. The approach chosen in this work is a fused multiply add architecture [76]. This will be described further in Section 6.2.

6.1.2 Activation Function Generator

Given its desirable non linear characteristics and ease of differentiability, a sigmoid based activation function, such as that defined in Eqn. 6.1, is commonly used in ANNs [158]. However, a direct hardware implementation is not practical as it requires excessive logic and also results in significant power loss.

$$y(x) = \frac{1}{1 + e^{-x}} \tag{6.1}$$

Consequently, a number of approximations amenable to hardware implementation have been developed. For ease of implementation reasons, LUT based schemes are frequently used, however the approach is costly in terms of area and power compared to the approximation precision achieved. Alternative approaches typically fall into the following broad categories: CORDIC, piecewise linear approximations [203][204][205][206] [207][208], piecewise second order approximations [206] and combinatorial input/output mappings [209]. Furthermore, there is considerable variance within each category. For example, an A-Law companding technique is used in [203], a sum of steps approximation is used in [204], a multiplier-less piecewise approximation is

presented in [205] and a recursive piecewise multiplier-less approximation is presented in [208]. An elementary function generator capable of multiple activation functions using a first and second order polynomial approximation is detailed in [206]. Recently, a combinatorial approach has been suggested that considerably reduces the approximation error [209]. The approach chosen in this research, which is described in detail in Section 6.2, uses a first order minimax polynomial approximation. The use of a minimax polynomial has been suggested before in the context of a floating point activation function approximation [207], however the proposed approach further minimises the maximum error and implements an area and power efficient architecture. As will be seen from the benchmarks in Section 6.4, this approach produces the best approximation error, whilst being suitable for the implementation of multiple activation functions.

6.1.3 Number System and Precision Requirement Considerations

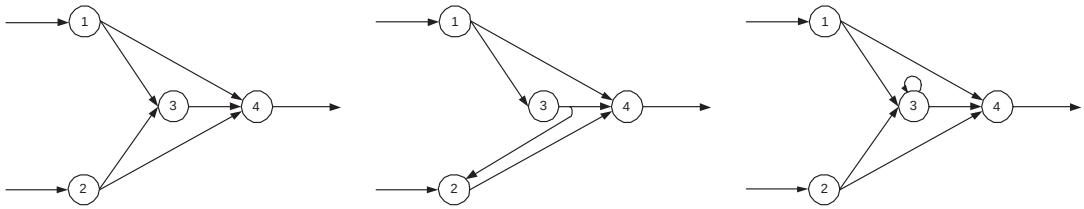
A fundamental design decision for any digital hardware architecture, particularly in the absence of a standard, is the choice of a number system and the associated word length. These have important consequences on the achievable range, precision and resolution of the final system. Furthermore, the word length impacts the critical path in the digital design and the power consumption. Low complexity reduced wordlength ANN approaches have the benefit of allowing reduced area, but this typically comes at the expense of output performance and reusability. The consequence of an excessively reduced word length is that the ANN outputs can be completely different than those of a similar topology with a longer word length [210]. It is generally accepted that, as the resolution and precision of the inputs, weights and the activation function are reduced, so too is the ability of the ANN to act as a universal approximator [211][212]. On the other hand, a long word length format, such as the 32-bit IEEE single precision floating point, may be appropriate for representing the dynamic range of possible weight values, however not only is it excessive relative to tolerable system performance [212], it also results in additional logic switching and area requirement. Related to the wordlength selection, input/output standardising (sometimes called scaling) is advised in most cases [167]. Standardising the inputs results in a smaller integer range.

Wust et al used a 5-bit reduced floating point format for a number of ANN applications [213]. This format used one sign bit, one mantissa bit and three bits for the exponent and was coupled with a hybrid floating point / fixed point arithmetic core. However, he acknowledged that more bits would be necessary to support the higher precision required for on-chip training. Holt et al showed

that a ANN trained via back propagation required 16-bits (configured as one sign bit, 3 integer bits and 12 bits of fractional precision) for training and forward evaluation for difficult convergence problems [212]. Floating point and fixed point hardware ANN implementations were investigated from a area and throughput perspective by Moussa et al [182]. They concluded that the fixed point was 12 times faster and 13 times smaller than the equivalent floating point implementation. However, it is debatable whether this was a fair comparison as they used a fixed point format with 16 bits and compared this to a 32-bit IEEE compliant single precision floating point format. It is no great surprise that the larger word length format required more area and had a lower maximum frequency. A more enlightening comparison of floating point and fixed point implementations was recently carried out by Savich [210]. This work explored 25 different word length sized fixed point and floating point representations. They found that for comparable fixed point and floating point formats (i.e. comparable in the sense that the two formats gave similar precision), a fixed point representation typically required half the area of the floating point format, achieved a higher maximum clock frequency and had a shorter latency. Based upon these results, it is reasonable to also conclude that the energy consumed in a datapath dominated ANN architecture will also be smaller for a fixed point representation compared to a floating point format of comparable precision.

A stochastic representation is a further alternative number system sometimes used for hardware ANNs. Using this system the value of a signal in a digital ANN implementation is encoded using the probability of the given bit in a stochastic pulse stream being a logic one [214]. This has the benefit that many common arithmetic operations require very simple logic [215]. Whilst beneficial from an area perspective, there are issues concerning representation variance. Furthermore, due to the reduced throughput from the inherently serial operation, a higher clock frequency is required to match the throughput of a more parallel fixed point implementation. The substantially increased clock frequency is of considerable concern in the clock tree network from a power consumption perspective. The clock tree network can account for a considerable proportion of the total power consumption, particularly in deep sub micron technology processes.

From the observation of prior research, 16 bits in a 1.3.12 fixed point representation (one sign bit, three integers bits and ten fractional bits) is used throughout the proposed design, with the input data standardised to a range of -1 to 1. The additional integer bits allows the EANN accumulation values to grow to levels which maximally exploit the resolution achievable from the proposed activation function generator.



(a) Forward connections only

(b) Mixed connections: a recurrent connection exists between node 3 and node 2

(c) Mixed connections: a looped current connection exists at node 3

Figure 6.2: Neuron connection types

6.2 Proposed Hardware Architecture

Unlike a feed forward ANN, an evolved network can have any topology, potentially with a mixture of forward connections (see Fig. 6.2(a)), recurrent connections (see Fig. 6.2(b)) and looped current connections (see Fig. 6.2(c)). Furthermore, depending on the application and evolutionary process, the neuron connections could vary from being fully connected to being sparsely connected. Owing to the complexification process (see Section 3.3.4), NEAT will naturally favour sparse topologies. These factors have important consequences for the hardware architecture. The efficiency of systolic array architectures is dramatically reduced when operating on sparse neural topologies. This is because the data flow through the systolic array is frequently interrupted and thus the throughput benefit of multiple PEs is not achieved. To overcome this, the PE can be either disabled for that weighted sum calculation or have a weight of zero. However both solutions are undesirable. Disabling the PE leads to additional control logic for each individual PE. Whilst storing weights with a zero value leads to increased memory sizes, which further exacerbates power consumption issues, particularly as memory power consumption disproportionately increases with size. The systolic array efficiency is further reduced due to the presence of recurrent connections. This causes unpredictable feedback connections from other neurons, causing further data flow interruptions. However more importantly, as systolic array architectures favour a layered ANN, where the inputs to each PE layer are well defined, it is not a trivial matter to dynamically reconfigure the PE inputs for the evaluation of alternative topologies, which contain recurrent links. This situation would be necessary for an application where NEAT is evolving in real-time, such as artificial intelligence for computer gaming [31]. Whilst it is possible to modify the genetic algorithm so that only forward connections are added, this compromises the quality of the evolved solution.

These factors have provided motivation for the research in this thesis to explore alternative ar-

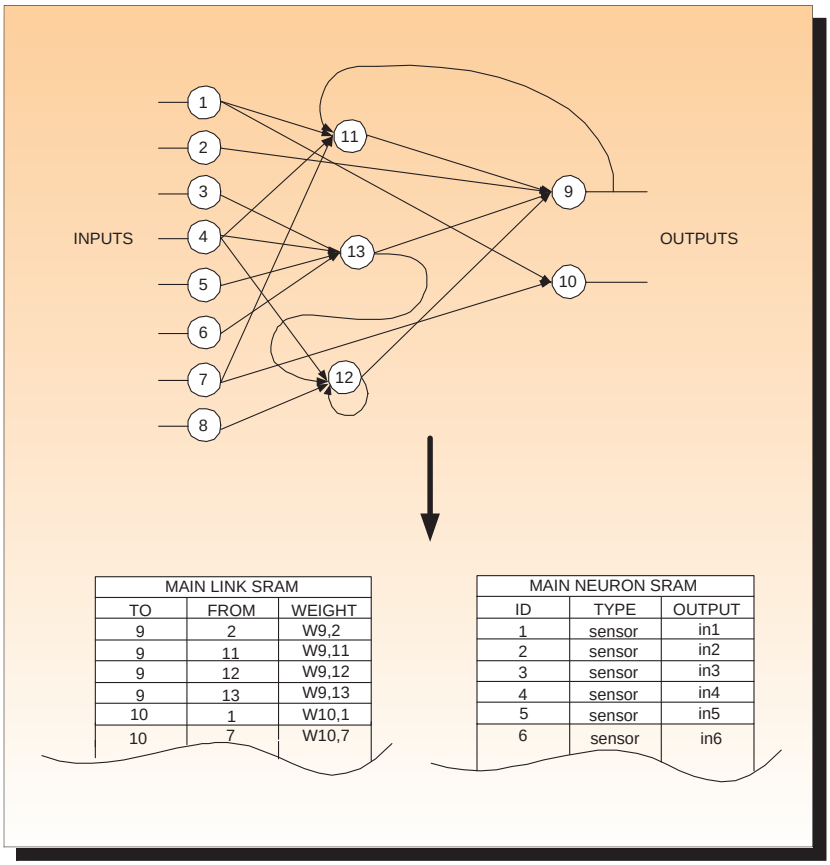


Figure 6.3: NEAT phenotype to hardware memory mapping

chitectures rather than a systolic array approach. Examining the NEAT phenotype data structures, it is clear that the essential information in the link and neuron genes, can be mapped easily to hardware memories, as is demonstrated from the simple phenotype in figure 6.3. Essentially the proposed approach "parses" through the *LINK* memory to retrieve the relevant weights and using the "*LINK→FROM*" field as the index to retrieve the output from the appropriate neuron in the *NEURON* memory. Once the values are retrieved the multiply accumulate operation is performed. This operation is repeated for all connections associated with that neuron, before the activation function is calculated. The process then repeats for all neurons in the topology. This approach is analogous to how a given topology would be evaluated on a single-core processor in software. However, the proposed hardware architecture exploits the inherent parallelism that exists in the topology to increase throughput. To achieve this goal, two PEs operating in parallel are used, as can be seen from the proposed architecture in Fig. 6.4. The PE datapath is 64 bits and each PE has access to the memory (via a memory controller). Two PEs were chosen so as to give a good trade off between throughput and bus addressing complexity. Each PE is equipped with a local "*LINK*" SRAM. This is loaded with the incoming connections for that neuron. To reduce stalling,

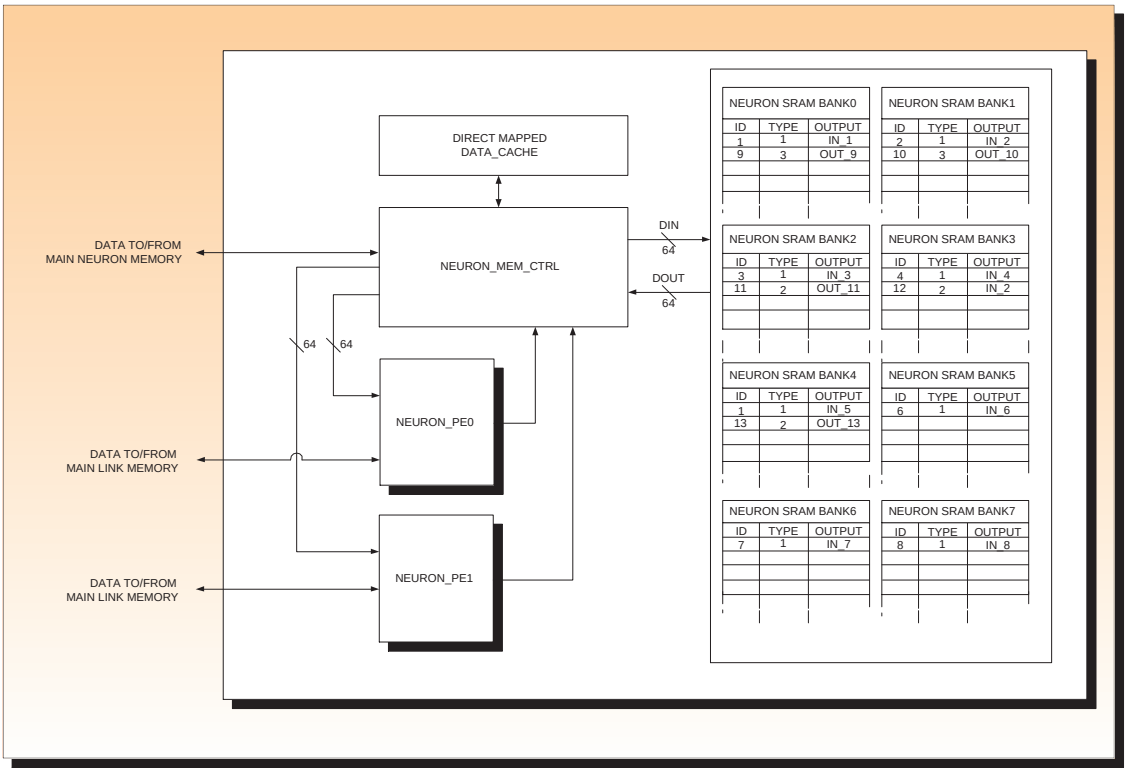


Figure 6.4: Simplified block diagram of the proposed EANN hardware accelerator datapath

the connections for neuron $N + 1$ are prefetched, whilst in parallel the PE processes neuron N. The PE datapath is 64 bits wide because 4×16 bit entries from the local *LINK* SRAM are retrieved in a burst and processed in parallel in each PE. This parallel processing is possible since the "*LINK→WEIGHT*" values are processed sequentially and they do not have interdependencies. The decision to use four parallel arithmetic units per PE was chosen as a trade off between throughput, bus width size and to minimise the complexity of retrieving multiple data units per clock cycle from the hierarchical memory system.

Unless a fully connected topology is being evaluated, the four "*NEURON→ID*" addresses decoded from "*LINK→FROM*" will not necessarily be contiguous. Therefore if a single "*NEURON*" SRAM is used (such as in Fig. 6.3), only one "*NEURON*" address could be processed per clock cycle. However to maximise the performance of the multiple arithmetic units in the PE datapath, valid input data needs to be available on each clock cycle. To overcome this issue, repartitioning the neuron memory into eight smaller SRAMs is proposed. This memory partitioning can be seen in Fig. 6.4. Eight SRAMs have been chosen as a consequences of selecting 2 PEs which each request 4 data values in parallel. To further increase the probability that all data units can be retrieved within one clock cycle, thus maximising the parallel processing potential, a data cache is used to maximise data reuse. The cache provides backup should two or more data requests occur

for the same "*NEURON*" SRAM bank. This could occur based on the connectivity of the topology, in particular if the sparse synapses were modulo eight apart.

When the memory control logic receives a request for 8 new values from the 2 PEs, the cache is firstly examined to see if it can fulfil these requests. Should cache misses occur, the "*NEURON→ID*" is decoded to indicate the relevant SRAM bank. If two or more requests attempt to access the same SRAM bank, the data must be retrieved over multiple clock cycles, otherwise all requests can be handled in one clock cycle. In the worst case scenario, when none of the data is present in the cache and all requested data is located in the same "*NEURON*" SRAM bank, one multiply accumulate operation occurs per clock cycle. On the other hand, if all the requested data is either in the cache or different Neuron SRAM banks, eight multiply accumulates occur per clock cycle. For a fully connected feed forward ANN, the throughput performance of the proposed solution will be identical to a similar sized systolic array (assuming the pipelining of the systolic array is full), although the memory bandwidth used by the systolic array will be smaller. As discussed earlier in this section, the throughput of the proposed solution will outperform a systolic array architecture when processing a mixed forward and recurrent sparse connection topology.

6.3 Implementation of Proposed Architecture

This section describes the implementation details of the proposed architecture. As shown in Fig. 6.4, the proposed architecture can be considered to consist of multiple PEs, control logic and a memory subsystem. The function of the PE is to generate the weighted sum of inputs to the neuron. The author has also chosen to add the activation function generator to each PE (see Fig. 6.5). An alternative approach is to time share a single activation function generator between multiple PEs. This is typically the approach adopted when using a systolic array architecture due to the highly regular processing. If such a time-sharing scheme was used in the proposed architecture, control logic must be designed to ensure only a single PE has control of the activation function generator at any clock cycle. However as there are only two PEs and the area overhead for the activation generator is not considerable ($< 2,000$ gates), a design decision was made to integrate the activation function logic into each PE.

The following subsections describe the implementation of the activation function generator, PE control logic and the cache structure. It should be noted that the memory banks which were described in section 6.2 are not elaborated upon further, principally due to the direct implementa-

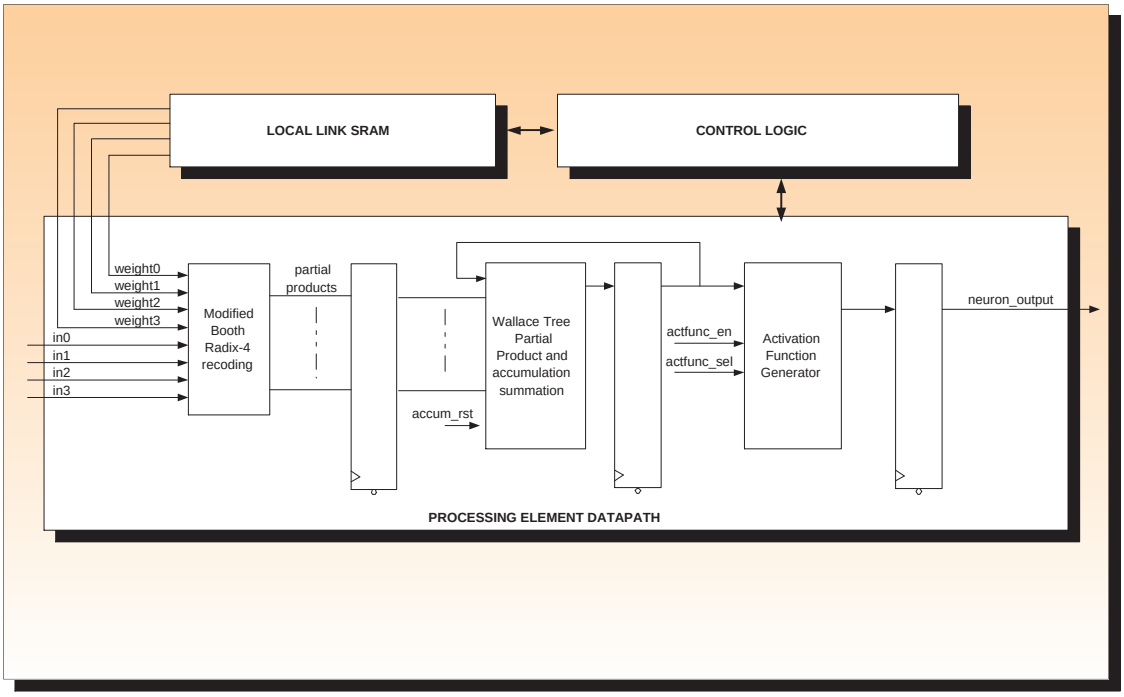


Figure 6.5: Simplified block diagram of the Neuron PE

tion that was used. Extra emphasis is placed upon the description of the activation function generator. In part this is because the implementation of the chosen activation function approximation scheme relies heavily on a multiply accumulate operation. Therefore, this multiply accumulate sub-block can be reused (albeit with slight modifications) to generate the weighted sum of inputs to the neuron activation function.

6.3.1 Activation Function Generator

A review of common function approximation schemes, including LUTs, CORDIC, approximating polynomials and splines, was given in Section 6.1.2. With the goal of achieving a good trade off in terms of area, approximation accuracy and ease of supporting multiple activation functions (desirable in an EANN if the selection of the activation function is under the control of the evolutionary process), a polynomial spline based approach was chosen by the author for further investigation. Polynomial approximating functions, the best known of which is the Taylor's Series, can be used to represent any arbitrary continuous function. Whilst the error in a Taylor's Series is very small at the expansion point, it rapidly increases at the boundaries of the interval (see Fig. 6.6). Therefore when using a finite order polynomial, often a more evenly distributed error is preferable. Consequently other approximating polynomials, such as least squares, have been employed [206]. This thesis proposes using a minimax approximation polynomial for the generation of the acti-

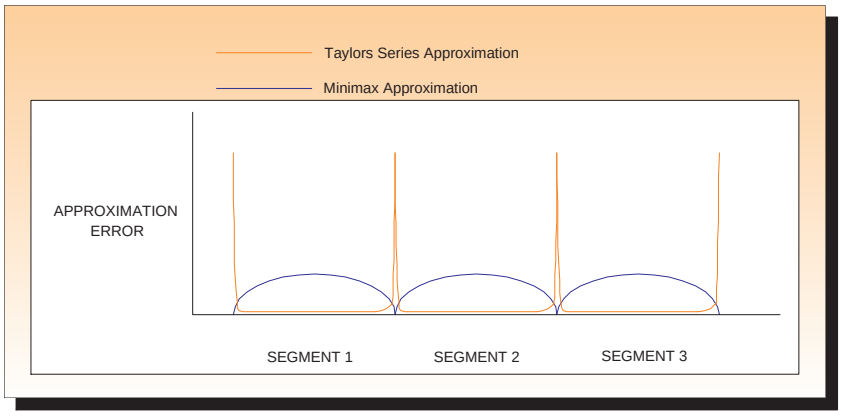


Figure 6.6: Error of Minimax Spline versus a Taylor Series Spline

vation function output [216]. A minimax polynomial exists for every approximation and has the characteristic of minimising the maximum error, by evenly distributing the error across the entire approximation range. To reduce the order of the approximating polynomial, the input domain of the function can be sub-divided into smaller intervals. This allows a polynomial of much lower order to approximate each of the sub intervals. The resulting composite function is known as a piecewise polynomial or spline. Using a spline-based activation function approximation offers the benefit that multiple activation functions can be accommodated with ease, by merely changing the coefficients of the approximating polynomial. This also means that minimal extra hardware is required to support the additional functions.

Using a Remez exchange algorithm within Matlab, appropriate coefficients were found to generate minimax polynomials on discrete intervals for each activation function [217]. The Remez exchange algorithm finds these coefficients by iteratively solving sets of linear equations. A first order minimax polynomial was used, as this has the benefit of avoiding higher order x^n operations. The disadvantage of using a lower order polynomial is that a greater number of splines are required to achieve the same approximation accuracy as a higher order polynomial. An increased number of splines results in a need for additional storage of the coefficients associated with each spline. Fortunately, in the attended application within this research, the effect of this property is not an issue, as the minimal error representable from 10 bits of precision (see Section 6.1.3) is reached when using only a small number of first order minimax approximating polynomials. To further reduce the number of polynomials required to achieve a specified accuracy, the common approach of range reduction is leveraged. Range reduction exploits inherent function redundancies such as function symmetry, periodic behaviour etc. to allow fewer polynomial segments represent the function.

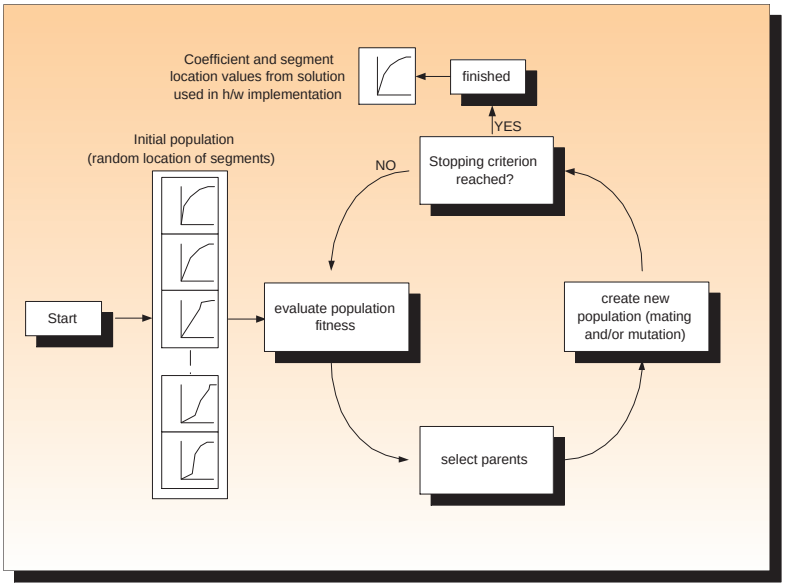


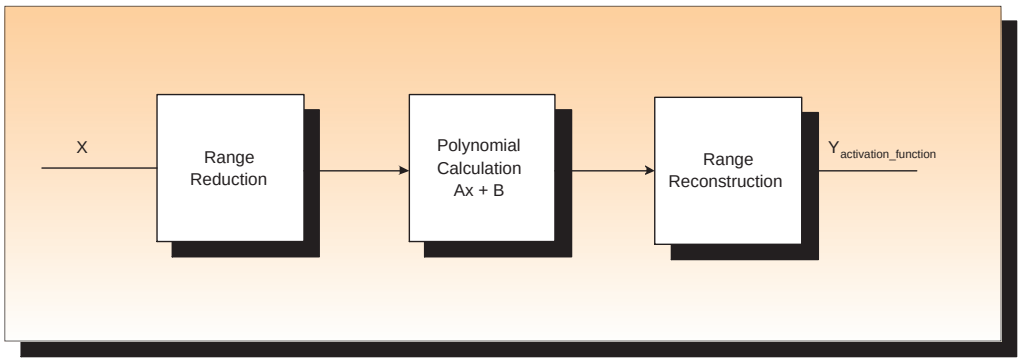
Figure 6.7: Genetic algorithm used in optimising location of spline knot points

The placement of the approximating polynomials on the input range clearly has a major bearing on the overall approximation error. The simplest approach is to evenly distribute the polynomials over the approximating range, although, astute placement can reduce the approximating error. The challenge of placing the polynomials in non-evenly distributed locations is that the potential search space is large. For example, when using 5 polynomials in a 0 to 8 range with a precision of 10^{-3}, there are in the order of 10^{17} possible combinations for the location of the polynomials. Due to the large search space issue, a GA was used to find the optimum location of the approximating polynomials. Unlike an exhaustive search, this solution is scalable even if the input range becomes extremely large, for example if using a double precision floating point representation. The GA was implemented using the Genetic Algorithm Optimisation Toolbox (GAOT) for Matlab [217]. The fitness function firstly uses the Remez exchange algorithm to generate the minimax polynomial coefficients for each candidate in the seed population. Using these coefficients, the minimax spline approximation for the chosen activation function (e.g. sigmoid) is constructed. The mean and maximum errors are then calculated from this approximation (using at least 10^6 samples) and a proportionate fitness is assigned to each candidate solution in the population. The GA selects suitable candidates for further evolution and generates a new population from these using crossover and/or mutation. As shown in Fig. 6.7 the process repeats until a stopping condition (e.g. a target average and/or mean error) is reached. As is the norm, the GA needed extensive tuning through trial and error exploration of the different input parameters. Initial population sizes of 10 to 1,000 were considered, along with extensive investigation into different crossover functions

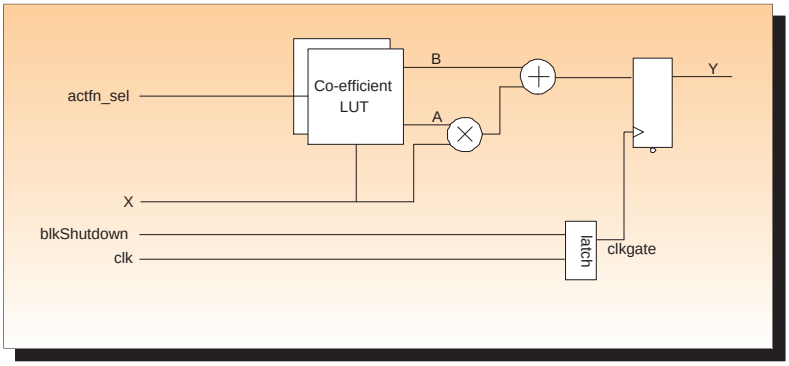
and different mutation functions. The GA achieved best results using arithmetic crossover and a multi-point non-uniform mutation with 5 mutation points. The GA typically improved the approximating error by 30% to 60% relative to an even distribution of the approximating polynomials. A full comparison of the proposed approximation scheme against prior art is given in Section 6.4.

The outputs of the GA optimisation phase are a pair of A and B coefficients for each linear spline-based polynomial, as well as the optimal location of each spline on the input range. This fixed information is then used in the hardware implementation. A simplified block diagram of an associated datapath to calculate the activation function approximation can be seen in figure 6.8(a). The range reduction and range reconstruction blocks for a sigmoid activation function have a trivial implementation, because only symmetry around the Y-axis can be exploited. Nevertheless, range reduction allows the coefficient storage space to be halved. Figure 6.8(b) shows a direct implementation of the linear polynomial ($AX + B$) calculation. The value of the input, X, is used as an index into a LUT, which stores the values of the A and B coefficients.

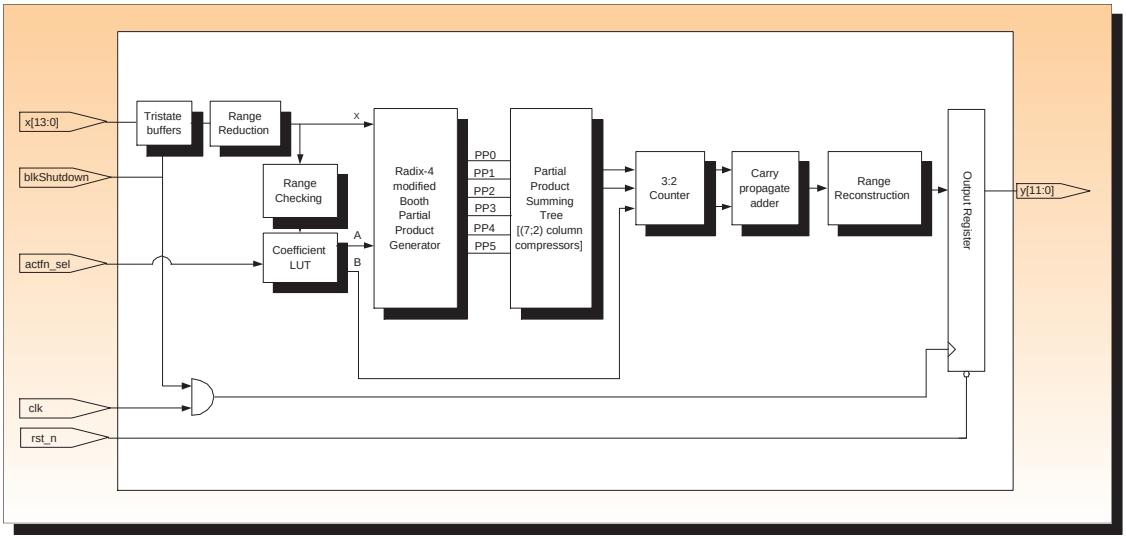
A direct implementation of the polynomial calculation uses a separate multiplier and adder. However, as there is redundancy in the multiply accumulate operation, the operations can be combined to reduce area and switching. This architecture is commonly known as a fused multiply add and can be seen in Fig. 6.8(c) [76]. The number of partial products has been halved by using modified Booth radix-4 coding. The addition of the partial products is calculated using (7;2) column compressors (see Fig. 6.9(a)) [76][218]. As the carry out of the (7;2) compressor does not depend on the carry in, the critical path is improved and a higher clock frequency is supported (see Fig 6.9(b)). Furthermore, the generation of the two's complement for the modified Booth algorithm uses a simple inversion for the one's complement and delays adding the additional one, until the partial product accumulation, thereby reducing the critical path. The resultant carry and save are added to the B coefficient using a (3:2) counter. The final sum and carry are then added using an efficient carry propagate adder. Employing these techniques provides the proposed activation function architecture (see fig. 6.8(c)) with a good trade off between speed, area and power consumption. The architecture is easily modified for additional pipeline stages should a higher throughput be required. Power reduction can be further tackled at the gate level, by employing clock gating and using tristate buffers on the inputs. These two techniques help minimise the power when the block is not active.



(a) Simplified datapath of activation function generator

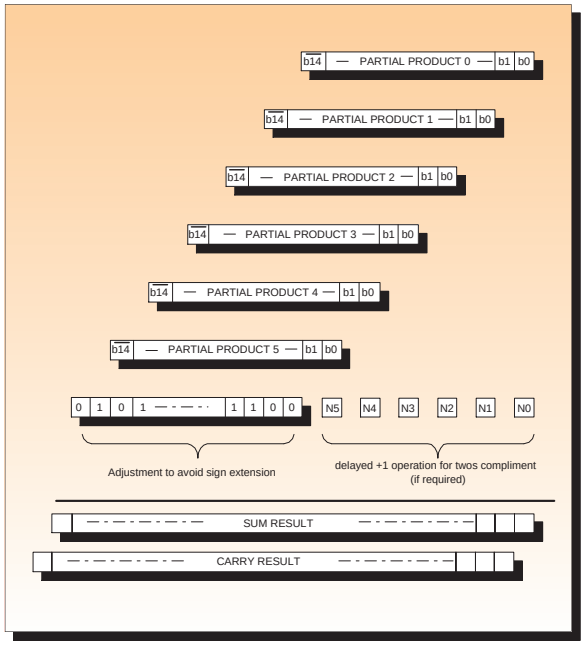


(b) Polynomial evaluation

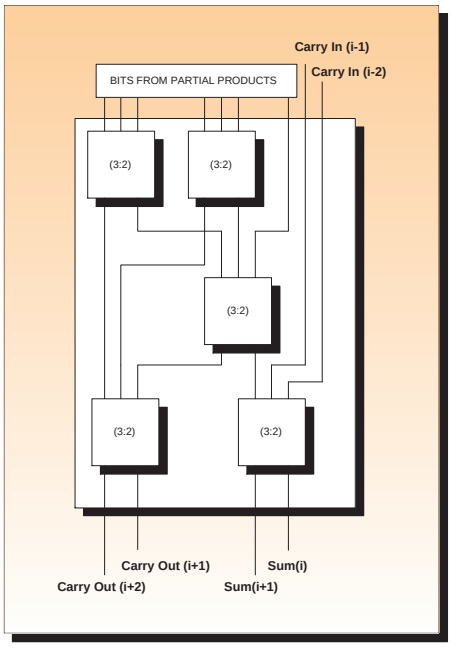


(c) Proposed hardware architecture for activation function generator

Figure 6.8: Implementation of Activation function generator



(a) Partial Product Summing Tree



(b) 7;2 Column Compressor

Figure 6.9: Elements of the Efficient Multiply Accumulate Hardware

6.3.1.1 Remaining Neuron PE logic

The weighted sum of inputs to each neuron is implemented using a similar architecture to that which was used for the multiply accumulate in the activation function generator (see Fig. 6.8(c)). Saturated arithmetic is used to prevent the arithmetic from wrapping around and giving incorrect answers should input values give a result outside the bounds representable from the chosen wordlength. Each PE has $\approx$ 3KB of local SRAM to store "*LINK→FROM/WEIGHT*" data, providing enough storage for the details of over 1000 neuron connections. Obviously the amount of memory can be adjusted based upon the timing constraints of the main memory and connectivity characteristics of a particular application. The PE control logic, which governs access to the local SRAM and the control signals for the PE datapath is outlined in algorithm 6 below.

Algorithm 6: Neuron PE datapath control flow

1 ***MEM_SETUP setup***;
load PE SRAM with "LINK→FROM/WEIGHT" data associated with the first "LINK→TO" neuron;
Regular processing starts once loaded;
In parallel with processing, prefetch "LINK→FROM/WEIGHT" data for the next neuron and load into PE SRAM;
2 ***LINK_DECODE Stage***;
PE requests 4 "LINK→FROM/WEIGHT" entries from local SRAM;
3 ***INPUT_FETCH Stage***;
Using the 4 retrieved "FROM" addresses, the PE requests these values from the "NEURON" SRAM memory banks;
The "WEIGHT" values are set up on the inputs to the partial product generation logic;
4 ***PP_GEN Stage***;
With the input values returned from the "NEURON" SRAM banks, the partial product generation logic is enabled;
5 ***ACCUM Stage***;
The partial products and the previous accumulation value are added in the Wallace Tree compressor;
6 ***ACT_FN Stage***;
If necessary the activation function generator is enabled;
7 ***WRITE_BACK Stage***;
If the activation function was enabled, the output is written to memory during this clock cycle;

6.4 Results

This section benchmarks the performance of the proposed activation function approximation scheme as well as discussing synthesis results and power consumption estimates of the overall EANN architecture. The architecture implementation results are principally compared against prior ANN hardware implementations. It should be noted that a fair comparison is challenging, as no other approach from the outset attempts to accelerate the ANN phenotype evaluation within EANNs and exploit the associated sparse topologies. In as much as possible, the benchmarking metrics are normalised to attempt to account for this issue and other implementation features of each

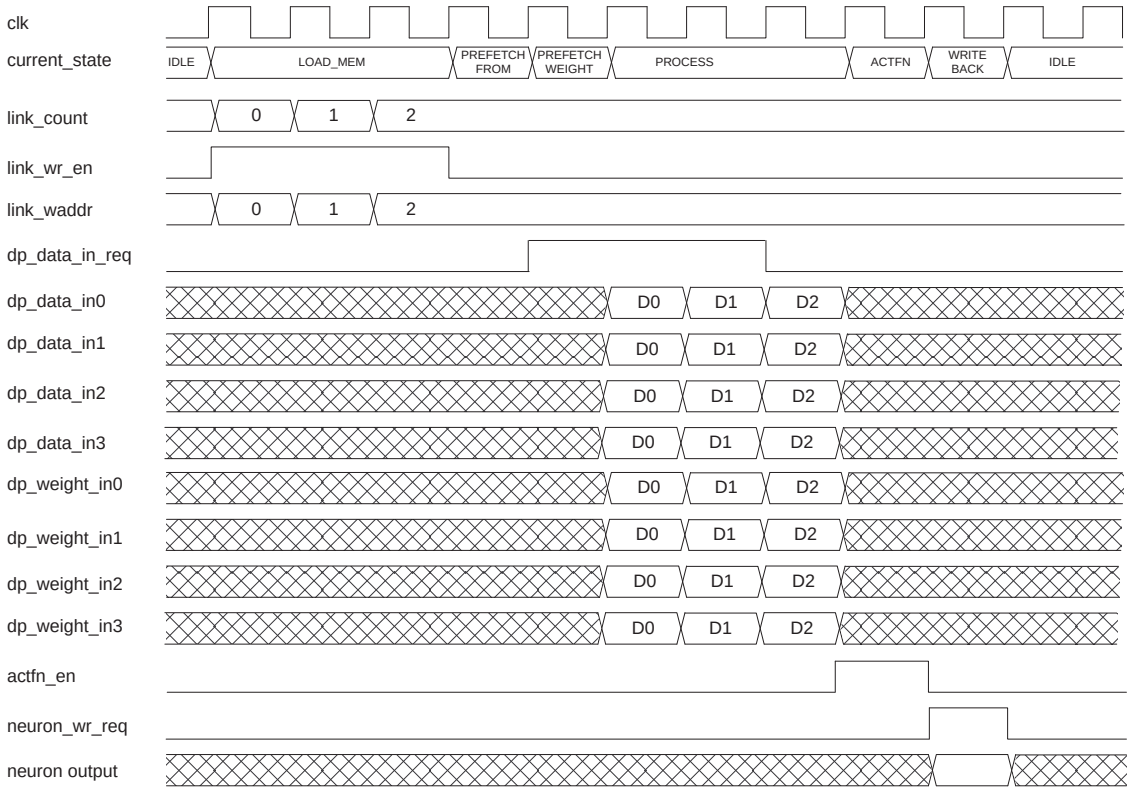


Figure 6.10: Neuron PE timing diagram

benchmarking architecture (e.g. different maximum clock frequencies, different semiconductor technology processes). As power consumption estimates are rarely given in the hardware ANN literature, for the purposes of comparison, the generated power estimates will be compared against the power consumption required in a software implementation. The software power consumption estimates were generated and discussed in Section 5.2.1.

6.4.1 Activation Function Results

Due to its widespread usage, the sigmoid activation function approximation was used to compare the proposed activation function approximation scheme against prior research in the literature. Using a sigmoid activation function the proposed approach was evaluated using 2 to 8 approximating polynomial segments. The maximum and average error results for these tests can be seen in table 6.1. This table allows a fair comparison of method error between various approaches who share the same range and number of segments (relevant for piecewise approximations). In the cases where a direct comparison is possible, the proposed approach clearly outperforms related research. Tommiska uses a direct input/ output combinatorial mapping, therefore the number of segments is not applicable [209]. In this case it is fair to claim the proposed approach outper-

Table 6.1: Maximum and average Sigmoid Approximation errors

Design	Range	Segments	Maximum Error	Average Error
Myers et al [203]	[-8,8)	N/A	0.0490	0.0247
Alippi et al [204]	[-8,8)	N/A	0.0189	0.0087
Amin et al [205]	[-8,8)	3	0.0189	0.0059
Vassiliadis et al (First Order) [206]	[-4,4)	4	0.0180	0.0035
Vassiliadis et al (Second Order) [206]	[-4,4)	4	0.0180	0.0026
Faiedh et al [207]	[-5,5]	5	0.0050	n/a
Basterretxea et al (q=3) [208]	[-8,8)	N/A	0.0222	0.0077
Tommiska (337) [209]	[-8,8)	N/A	0.0039	0.0017
Tommiska (336) [209]	[-8,8)	N/A	0.0077	0.0033
Tommiska (236) [209]	[-4,4)	N/A	0.0077	0.0040
Tommiska (235) [209]	[-4,4)	N/A	0.0151	0.0069
Proposed approach	[-8,8)	2	0.0158	0.0068
Proposed approach	[-8,8)	3	0.0078	0.0038
Proposed approach	[-8,8)	4	0.0047	0.0024
Proposed approach	[-8,8)	5	0.0032	0.0017
Proposed approach	[-8,8)	6	0.0023	0.0012
Proposed approach	[-8,8)	7	0.0017	0.0009
Proposed approach	[-8,8)	8	0.0013	0.0009

forms Tommiska's approach on an ± 8 range when using 5 or more segments. The improvement achieved by leveraging the effective search capabilities of genetic programming is apparent from comparing the five segment evaluation of the proposed approach with the five segment evaluation of Faiedh et al [207], who employs a minimax polynomial with a floating point data representation. The GA could be considered to be predominately responsible for the 36% improvement in error. Table 6.2 shows maximum and average approximation errors for additional functions, including Tanh and the sigmoid derivative. Tanh is another commonly useful activation function, whilst the sigmoid derivative is important if on-chip training (using back propagation) is used. The proposed approach gives a better maximum error than both the first and second order approximation of Vassiliadis et al [206]. It is reasonable to conclude that the proposed activation function approximation scheme compares favourably with the state of the art literature. It achieves high precision and has the benefit of easily supporting multiple activation functions, should these be required during the evolutionary search process.

6.4.2 Hardware Implementation Results

The hardware design was captured in Verilog HDL and synthesised using Synopsys Physical Compiler for a 90nm TSMC ASIC library. Physical Compiler was also used to generate an automatic

Table 6.2: Max and Mean Errors approximations for other functions

Design	Sigmoid Derivative			Tanh		
	Range	Maximum Error	Average Error	Range	Maximum Error	Average Error
Vassiliadis et al (1^{st} Order)	[0,8)	8.8×10^{-3}	2.6×10^{-3}	[0,8)	5.7×10^{-2}	5.0×10^{-3}
Vassiliadis et al (2^{nd} Order)	[0,8)	4.6×10^{-3}	5.0×10^{-4}	[0,8)	1.6×10^{-2}	1.6×10^{-3}
Our approach (4 segments)	[0,8)	3.1×10^{-3}	1.7×10^{-3}	[0,8)	9.5×10^{-3}	2.4×10^{-3}

placed and routed implementation (using minimal physical constraints). The netlist was back annotated with the delay information (SDF file generated from Synopsys PrimeTime) and layout parasitic information (SPEF file generated by Physical Compiler during the automatic layout flow). A gate-level simulation was then run (in Synopsys VCS 7.2 Simulator) to generate realistic gate switching. The switching information was saved in a VCD file, which along with the gate-level netlist was used as inputs to Synopsys PrimePower to generate a power consumption estimate using a 1.2V voltage source. The design flow adopted assumed conservative worst case operating conditions. It should be noted that memory synthesis and layout was excluded from the ASIC design flow due to lack of appropriate memory compiler tools. However, this memory was prototyped in a Xilinx FPGA design flow.

The target clock frequency chosen was 250 MHz, the motivation for this was driven by the following throughput calculations. Firstly, an assumption was made that a minimum sized face should be 24x24 (meaning the top layer of the image pyramid is skipped) and the input image/frame size is CIF-sized (352x288). Inspecting all locations in the subsequent image pyramid (assuming a subsampling factor of 1.2), requires 174,041 20×20 pixel block sized ANN classifications (see Eqn 6.2). Using the assumption that real-time video performance would require 25 frames per second, this means 4,351,025 ANN classifications per seconds. Therefore, the time budget for a single ANN evaluation is approximately 230 ns. As the ANN processing time is dominated by the multiply accumulate operation, it is reasonable to conclude that the clock frequency will be proportional to the maximum number of connections in the trained topology, which needs processing within the time budget of 230ns. The largest trained topology in Chapter 4 required 411 connections and the proposed architecture allows 8 multiply accumulates per clock cycle (under best case conditions). Therefore the minimum clock frequency to allow real-time video performance without skipping locations (e.g. via skin detection etc) is $\frac{230ns}{(411/8)} = 4.47$ns. Allowing a 10% margin of safety (e.g. to allow pipeline stalling if the output of a neuron is required during the clock cycle in which it is being updated) the clock period used was 4ns or a frequency of 250 MHz. To process ensembles of NEAT based classifiers within the same time period, it is simply a

matter of creating multiple instantiations of the proposed architecture.

$$
\begin{aligned}
(293-20)\times(240-20) &= 60060 \text{ - Layer 2 of image pyramid}\\
(244-20)\times(200-20) &= 40320 \text{ - Layer 3 of image pyramid}\\
(204-20)\times(166-20) &= 26864 \text{ - Layer 4 of image pyramid}\\
(170-20)\times(138-20) &= 17700 \text{ - Layer 5 of image pyramid}\\
(141-20)\times(115-20) &= 11495 \text{ - Layer 6 of image pyramid}\\
(141-20)\times(115-20) &= 11495 \text{ - Layer 7 of image pyramid}\\
(117-20)\times(95-20) &= 7275 \text{ - Layer 8 of image pyramid}\\
(97-20)\times(79-20) &= 4543 \text{ - Layer 9 of image pyramid}\\
(80-20)\times(65-20) &= 2700 \text{ - Layer 10 of image pyramid}\\
(66-20)\times(54-20) &= 1564 \text{ - Layer 11 of image pyramid}\\
(55-20)\times(45-20) &= 875 \text{ - Layer 12 of image pyramid}\\
(45-20)\times(37-20) &= 425 \text{ - Layer 13 of image pyramid}\\
(37-20)\times(30-20) &= 170 \text{ - Layer 14 of image pyramid}\\
(30-20)\times(25-20) &= 50 \text{ - Layer 15 of image pyramid}\\
&\overline{\qquad\qquad\qquad\qquad\qquad\qquad}\\
&174,041 \text{ Total candidate face locations}
\end{aligned}
\tag{6.2}
$$

Using a clock frequency of 250MHz, the synthesis of the proposed architecture (excluding SRAM memories) gave a gate count of 79,607 gates. For purpose of comparison with prior art, the design was also targeted to a Xilinx Virtex-2 FPGA. Using the Xilinx ISE 8.2 design flow, the design required 1,650 slices and had a maximum clock frequency of 47MHz. However, it should be noted, that no FPGA optimisations were made to the RTL code nor were the on-chip multiplier resources used. Therefore the FPGA slice count offers considerable scope for optimisation.

Using the ASIC design flow, the power consumption was estimated at 12mW. For the reference trained topology with 411 connections and 17 neurons the processing required 57 clock cycles, resulting in 3.6nj of energy dissipation. It was shown in Chapter 5, that a software implementation of the algorithm required 662,260 clock cycles on an 120MHz ARM920T processor. This figure did include the overhead of the NEAT software library, therefore for a fairer comparison between

a mobile device software implementation and the proposed hardware architecture, the number of clock cycles required on the ARM 920T processor for the individual multiply accumulates and activation functions is used instead. Using the reference topology (411 connections & 17 neurons) this results in the following best case (i.e. control overhead is ignored) software energy dissipation:

$$(411 \times 4\mu s) \times 41mW + (17 \times 45\mu s) \times 49mW = 104.89\mu j \tag{6.3}$$

This figure cannot be compared directly with the energy dissipation of the hardware architecture due to the differing clock frequency rates (120MHz in the ARM920 as opposed to 250MHz in the proposed hardware architecture). Furthermore, as different supply voltages were used in the generation of each set of results, this must also be taken into consideration. The voltage and frequency of the proposed hardware architecture are adjusted theoretically to match those of the ARM-920T processor, thus allowing a fairer comparison[1]. By reducing the clock frequency to 120MHz, the power consumption will decrease by approximately a factor of 2 (see Chapter 2 for further details), but the processing time will also increase by a factor of two. Adjusting for the voltage difference (2.5V on the ARM 920T development board and 1.2V for the proposed hardware architecture) the power is increased by a factor of $\left(\frac{2.5V}{1.2V}\right)^2 \approx 4$ (see Chapter 2 for further details). Finally, as both implementations used 90nm semiconductor technology processes, there is no need to compensate for differing semiconductor process size. Therefore the scaled energy dissipation of the proposed hardware is:

$$3.6nj \times \left(\frac{2.5V}{1.2V}\right)^2 = 15.63nj \tag{6.4}$$

This means the proposed hardware architecture is $\frac{104.89\mu j}{15.63nj} = 6,711$ times or three orders of magnitude more energy efficient than a direct implementation of the proposed algorithm in software on an ARM 920T processor. Whilst it is acknowledged that further optimisation of the ARM code could reduce this figure, its reasonable to surmise that the proposed hardware architecture would remain at least two orders of magnitude more energy efficient. The order of magnitude of energy efficiency of the dedicated hardware architecture over a software implementation is consistent with that reported in similar hardware software energy efficiency benchmarking [37]. Overall, it is reasonable to conclude that the proposed hardware architecture compares very favourable

[1]When using a lower clock frequency and higher voltage the proposed hardware architecture will meet the revised timing constraints. The alternative scaling approach of theoretically decreasing the clock period and voltage for the ARM-920T solution will not be guaranteed to meet the timing constraints and as such could create an unstable system

against a software implementation in terms of throughput and energy efficiency.

6.4.3 Benchmarking of Proposed Hardware Architecture against Prior Art

Comparing the proposed hardware architecture to prior art is challenging for multiple reasons. Firstly, the author is not aware of any other architecture which specifically attempts to accelerate the EANN phenotype calculation. However, as the evolutionary training is processed offline in software, it is reasonable to compare the proposed hardware architecture purely against the significant amount of prior art in the field of hardware implemented ANNs. However, even when considering only the digital subset of hardware ANN research, direct unbiased comparisons are difficult. As well as the usual challenges encountered when attempting to compare hardware designs with different implementation environments and constraints (e.g. tool flow, ASIC/FPGA, different semiconductor process technology, latency, clock frequency, etc.), ANNs present further complications due to differing ANN architectures (e.g. topologies, activation functions, training algorithms) and vastly differing hardware architectural trade-offs (e.g. systolic arrays, SIMD-like datapaths, number of PEs, datapath wordlength). Furthermore, due to the diversity of applications in which ANNs are deployed, an optimum solution is likely to require considerable design trade-offs for a target application. For example, a highly parallel, high throughput ANN accelerator will be of little use on a mobile device if it consumes excessive amounts of power. Similarly, a low power, lower throughput ANN accelerator may also be of limited usefulness on a mobile device due to insufficient throughput processing capabilities. As previously stated, the goal of the proposed hardware architecture is to offer an energy efficient solution for (principally) semantic object image/video processing applications (in particular face detection) on mobile devices. The remainder of this section will demonstrate that the design choices made result in an architecture which compares favourably with prior art and offers appropriate trade-offs for the intended mobile device deployment target .

The proposed hardware architecture will be compared to the prior art in terms of throughput, silicon resource usage and where possible energy efficiency. This comparsion is evaluated against the small number of hardware ANNs for face detection, such as the architectures proposed by Theocharides et al, Sadri et al and Smach et al [143] [144][145]. To allow a more thorough comparsion, a selection of generic ANN hardware architectures are also used in the benchmarking process [184] [219][220][213][221] [197][222][192][185]. In the literature for benchmarking purposes, the speed/throughput of a hardware ANN architecture is frequently measured using the

Table 6.3: Benchmarking of Proposed Architecture against Prior Art

Design	Architecture	Word length	Implementation Process	Max. Clock Frequency	Silicon Resources	CPS	Power
Proposed	2 PEs each with 4 MACs	16	90nm ASIC	250 MHz	79,607 gates 0.123mm^2	2×10^9	12mW
			Xilinx Virtex-2	47 MHz	1,650 slices	376×10^6	
Kondo [184]	12-PE SIMD	24	0.5μm ASIC	50 MHz	97mm^2	300×10^6	4W
Theocharides [143]	Fixed logic	8	0.16μm ASIC	75.5MHz	30.4mm^2	n/a	7.35W
Schoenuer [219]	SIMD	16	0.35μm ASIC	109MHz	100K gates	n/a	2.5W
Sadri [144]	Fixed logic	9	Xilinx XC2VP20	200MHz	5x10^3 LUTs	n/a	n/a
Smach [145]	Fixed logic	8	Xilinx XCV1000	52 MHz	12.248x10^3 slices	n/a	n/a
Gadea [220]	10-PE SA	16	Xilinx XCV400	100 MHz	3,473 slices	81×10^6	n/a
Wust [213]	SIMD	5	Xilinx XC4020E	8 MHz	n/a	32×10^6	n/a
Lettnin [221]	Fixed logic	20	Xilinx XCV2000E	33 MHz	19,200 slices	n/a	n/a
Szabo [197]	Fixed logic	12	Xilinx XC4000	50 MHz	n/a	n/a	n/a
Popescu [222]	Fixed logic	8	Altera Flex10K	27 MHz	744 cells	n/a	n/a
Amin [192]	12	16	n/a	n/a	n/a	n/a	n/a
Denby [185]	SA	16	Xilinx XC2V8000	120 MHz	70% Utilisation	n/a	n/a

connections processed per second (CPS) metric and the connection updates per second (CUPS) metric for the recall phase and learning phase respectively. In the proposed architecture, the CPS metric is the more appropriate choice to evaluate the proposed design, as the evolutionary training will lead to a CUPS metric that has the same value as the CPS metric. As shown in Table 6.3, the proposed architecture compares favourable with the prior hardware implementations which quote the CPS metric. For example, the CPS metric gained from the proposed architecture is over 6 times greater than the architecture proposed by Kondo et al [184]. Although, different clock frequency, datapath wordlength and semiconductor technology processes are used in both cases. In this way, both the CPS and CUPS metrics can be a little misleading as they are not normalised across designs. For example, it is obviously easier to process connections that use single bit inputs/weights, compared to connections using 32-bit inputs/weights, which will have a far longer critical timing path. Furthermore, these metrics do not take account of the number of PEs or clock frequency. However, fully normalising these metrics for two designs, would show little distinction between the two, since such a normalisation would reduce to a measure of the speed of a single multiply accumulate, implemented using the same sized semiconductor process technology. The best that one can conclude is that the proposed architecture results in a throughput appropriate for image/video processing on a mobile device and compares very favourably to prior art, whilst acknowledging that a reimplementation of the prior art using similar semiconductor technology process would also yield a considerable speed improvement for those designs.

Ideally to compare the silicon resource usage of the proposed architecture against prior art, a technology independent metric such as the total gate count should be used, particularly for an ASIC implementation. In some instances in the literature, die size is quoted rather than gate count

and as such for completeness Table 6.3 lists the gate count and silicon die size that were recorded for the proposed architecture. Using both metrics, the proposed design compares favourably to the selection of digital ASIC solutions shown in Table 6.3. It should be noted, that a completely fair comparison of the die size area from two designs is only possible if the same semiconductor technology process technology is used for both designs. To a first order, it is reasonable to normalise these die area figures by the square of the ratio of the channel width relative to the proposed design (i.e. this assumes the channel length was scaled by the same ratio). With this scaling the architecture proposed by Kondo et al has a area of 2.9mm^2 and the design by Theocharides has a die size of 9.6mm^2. It should also be noted, that unlike the die size quoted by Kondo et al and Theocharides et al, the die area quoted of 0.123mm^2 for the proposed architecture, does not include memory. Furthermore, the datapath width is wider in the case of the design by Kondo et al and the figure quoted by Theocharides et al contains additional functionality for their face detection system (e.g. rotation detection). Nevertheless, it is reasonable to conclude that the proposed design compares favourably in terms of silicon compared to prior art which is implemented using ASIC technology.

For comparison against FPGA implementations, the proposed design was implemented using a Xilinx Virtex-2 FPGA. In this case the number of slices used by the design can be used for benchmarking purposes. It should be noted that the exact amount of logic that is contained within a slice is known to vary within each Xilinx FPGA product family, nevertheless it is a reasonable first order comparison metric. With the exception of Popescu, the proposed design is smaller than the other Xilinx FPGA implementations listed in Table 6.3. Compared to Popescu, more parallelism is exploited in the proposed architecture for higher throughput at a cost of increased FPGA resource usage. Again the issue of differing amounts of implemented functionality may lead to some distortions of the FPGA resource usage. However, overall it is reasonable to conclude that the proposed architecture compares very favourably with prior Xilinx FPGA implementations, particularly considering the proposed design was not optimised for a FPGA architecture. In light of multiple FPGA vendors with differing amounts of logic elements contained in a “primitive cell”, a potentially better way of comparing FPGA resource usage is via an “equivalent gate count” generated from each vendors tool flow. Unfortunately, this figure is not quoted for prior FPGA hardware ANN implementations.

For the intended image/video processing application on mobile devices, energy efficiency is vitally important. Table 6.3 shows that the proposed architecture compares very favourably against

Table 6.4: Normalisation of power consumption from related research

Design	Recorded Power	Power normalised to 90nm Process
Kondo [184]	4W	47mW
Theocharides [143]	7.35W	1.55W
Schoenuer [219]	2.5W	79mW
Proposed Architecture	12mW	12mW

the absolute power consumption estimates quoted in the prior art. Ideally, for purposes of fair comparison, normalising the power figures for differing clock frequency, supply voltage and semiconductor technology processes is required. For example, Kinane showed that for low power design the power P in a given process L with a voltage of V and a clock frequency of f can be normalised to a reference process L', voltage V' and frequency f' using the following formula [37]:

$$P' = P \times \left(\frac{L'}{L}\right)^2 \times \left(\frac{V'}{V}\right) \tag{6.5}$$

However, unlike the situations discussed by Kinane, the proposed design has an operating frequency equal to the maximum frequency. As there is little slack in the worst case timing path, the proposed design would inevitably fail timing constraints when using a larger process technology, thus making the normalisation in that direction invalid. Scaling the power consumption estimates from the prior art to the 90nm process used in the proposed architecture is feasible and is shown in Table 6.4. It should be noted that the clock frequency has not been scaled to match the clock frequency of the proposed architecture, as this is likely to cause timing path issues in a real design. Whilst it is difficult to make completely fair comparisons due to differing datapath wordlengths, clock frequency and exact amount of logic implemented, it is reasonable to conclude that the proposed design compares favourable to the prior art in terms of power consumption. Ideally, energy efficiency should also be compared. As well as allowing estimates of battery life, it would be beneficial in terms of fairer benchmarking, as the final energy figure would naturally take account of the levels of parallelism exploited in the design (i.e. this would be reflected in the execution time). Unfortunately, none of the related research in the literature quotes energy estimates.

6.5 Future Work

The proposed EANN hardware architecture could be improved in a number of ways. An obvious improvement is to share a single activation function generator between each PE, in this way the

overall silicon area could be reduced. In the situations where multiple PEs require access to the activation function generator logic, arbitration logic would be required to decide which PE should have access. Datapath stalling would most likely be necessary and the impact of this in terms of the throughput would require characterisation for different sized topologies. It would also be worth further investigating the reduced floating point and stochastic digital data representation formats. It may be possible to trade off an acceptable level of precision for reduced area and/or power. To improve the throughput, the obvious solution is to increase the number of PEs. However, the challenge with this approach is how to retrieve the data from bandwidth limited memories in order to maximise the processing potential of a high speed highly parallel datapath without the need for excessively expensive memories and to do so in a power efficient manner. This is an open research problem which would require exploration of alternative micro-architectures. As part of this investigation, the cache module also warrants further investigation, in particular the effects of different caches sizes, architectures when using different sized input topologies.

6.6 Summary of Contributions

The computational complexity associated with ANN/EANN processing for multimedia applications on mobile devices is highly undesirable from a throughput and power consumption perspective. This chapter has presented a viable hardware architecture to accelerate this processing, whilst also improving the energy efficiency compared to a software only implementation. Firstly in this chapter an overview of related hardware architectures for ANNs was presented. This included a discussion of activation function implementations and the available options for the overall data representation within a dedicated ANN architecture. In addition, the issues encountered when using a systolic array for ANNs with sparse topologies were elaborated. A detailed description of the proposed hardware architecture was then given. The architecture is flexible enough to process any ANN topology, whilst still providing a good trade off between accuracy, area, power and throughput. The architecture contains a novel activation function generator scheme which was optimised with a genetic algorithm. This scheme was subsequently shown in the results section to be an improvement upon related research. Efforts were made to fairly compare the proposed architecture against the prior art in terms of speed, area and power. Whilst this is a difficult and challenging task due to the diversity of the related research, the proposed architecture was shown to compare favourable. Furthermore, the proposed architecture was demonstrated to be consider-

ably more energy efficient than a software only implementation on an ARM920 processor. Finally a number of possible avenues of future research were outlined.

CHAPTER 7

Binary Motion Estimation Hardware Accelerator

In previous chapters, an EANN based face detection algorithm and associated hardware accelerator were described. One obvious way of using such a face detection segmentation mask is to encode that region with a higher bitrate or to improve the error robustness. Whilst this is very useful in itself, if the output of the processing is considered more broadly in terms of a semantic video object, it has the potential to facilitate a wider range of applications. For example, Chapter 1 highlighted that a semantic video object framework has the potential to allow novel applications throughout all phases of visual data processing, i.e video creation, storage, intelligent viewing/searching and transmission. As previously discussed in Chapter 2, MPEG-4 provides an object based framework for many of these applications already, in addition to providing a base on which to facilitate the development of novel video object based applications. One of the key aspects of MPEG-4 object based video coding (or indeed any future object based framework), is the separate processing of the video object shape information. From a run-time perspective, it was shown by Tseng et al that over 90% of the total resources required for MPEG-4 binary shape coding is consumed by binary motion estimation [45]. Although this task relates to the compression of the binary shape object, it is reasonable to conclude that such binary motion estimation for shape could also be useful/required for other novel semantic video object tasks/applications (e.g. shape tracking, automatic temporal upsampling of binary alpha plane etc.). As a consequence, the author believes that binary motion estimation is a vital reusable tool in a semantic video object processing “toolkit”, and as such requires efficient implementation for throughput and power con-

sumption reasons. The goal of this chapter is to address this need by proposing a novel hardware architecture for binary motion estimation. Firstly, a thorough review of hardware architectures for motion estimation is presented in Section 7.1, the proposed architecture is then described in Section 7.2 with associated implementation results outlined in Section 7.3. A discussion on potential avenues for future work is given in Section 7.4 and finally conclusions are presented in Section 7.5.

7.1 State of the Art Review

Motion estimation is acknowledged as the most computationally demanding task within video codecs. Depending on the codec configuration, motion estimation can use between 40% to 80% of the total resources required for the encoder. For this reason it is frequently implemented in hardware, particularly on computational constrained platforms. A generic hardware architecture for motion estimation (similar to that suggested by Kuhn [32]) is shown in Fig. 7.1. Local memory for the current macroblock and reference search window are used to reduce the activity on the main system bus and minimise the number of accesses to the main memory. When implementing a hardware architecture for motion estimation the principal design space considerations involve throughput, regularity, hardware efficiency, power consumption, I/O bandwidth and the quality of the prediction residual. In the past, with larger semiconductor technology process sizes the number of I/O pins was also a concern due to the likelihood of a motion estimation core taking up the full die and the die itself was likely to have strict pad constraints. For the most part with modern deep submicron technologies this could be considered to be no longer an issue, particularly as a motion estimation core is likely to be just a single element within a larger System on a Chip (SoC) design. Hardware implementations of the block matching algorithm have principally focused on systolic array architectures as this maximally exploits the full search algorithm, which is attractive from a hardware implementation perspective because of the regular data flow and the low control overhead. As a result, many hardware implementations use one dimensional (1-D) and two dimensional (2-D) systolic array architectures.

An early research effort by Yang et al used a 1-D systolic array architecture [223]. This architecture uses 16 PEs and on each clock cycle, one pixel from the current block and two pixels from the reference block are retrieved from memory. The pixel from the current block propagates through the design whilst the reference pixel is broadcast to each PE. The second reference pixel is

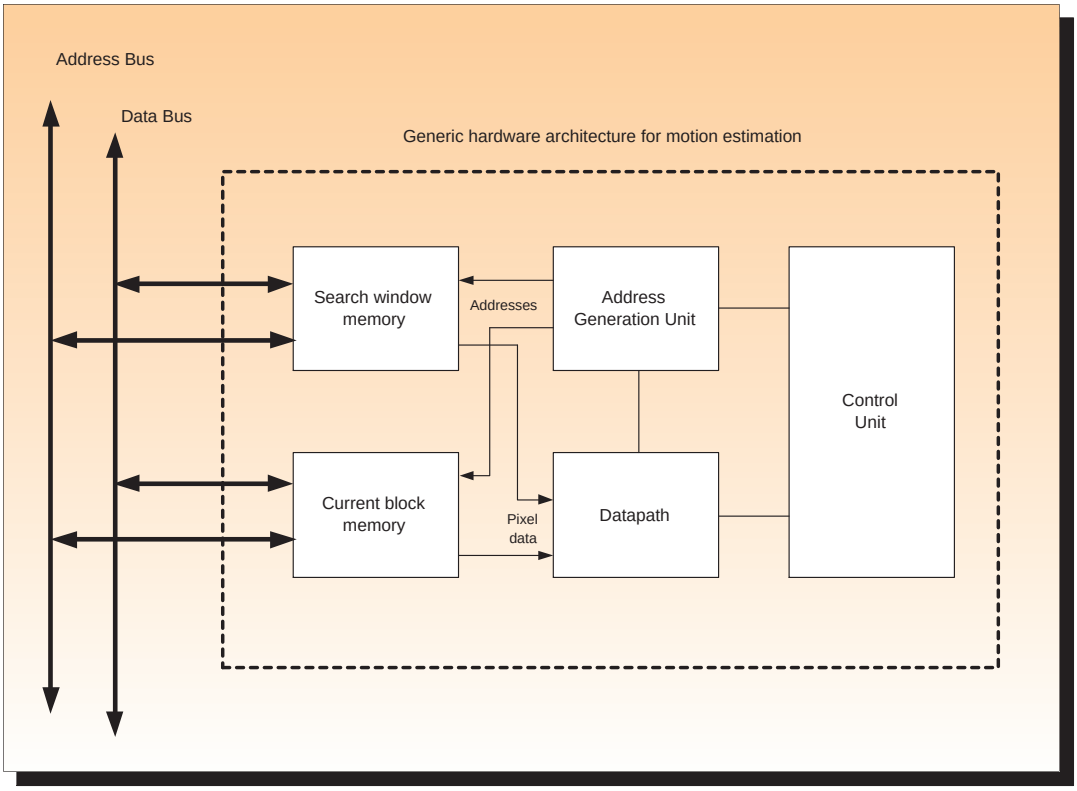


Figure 7.1: Generic hardware architecture for motion estimation

required when one or more PEs are operating on a different row of pixels. After 16 clock cycles the systolic array is filled and the first block match is produced after 256 clock cycles. A block match is then generated every clock cycle for the subsequent 15 clock cycles. For a $-7/+8$ vertical search window this process repeats 16 times in total. Thus this architecture generates a motion vector every 4111 clock cycles ($16 \times 256 + 15$). In the same era, Komaek and Pirch presented several architectures for 1-D and 2-D systolic array implementations for motion estimation [224]. Although the I/O bandwidth for the 1-D architecture proposed by Komaek and Pirsch is higher than that of the 1-D proposed by Yang et al. An input pin efficient 2-D systolic array architecture was proposed by Hsieh and Lin [225]. This architecture uses 3 pipeline stages, the first of which is the systolic array combined with shift registers, followed by a parallel adder and finally a "best match" selection unit. Chan and Panchanathan made a slight improvement on this architecture by removing the need for the parallel adder [226]. Jehng et al devised a tree based architecture [227], which whilst giving 100% PE efficiency has lower throughput than the architecture proposed by Hsieh and Lin, although the architecture has the benefit that it is reusable for the three step search strategy.

It is worth noting that hardware architectures for fast heuristic search strategies are possible

but introduce additional control logic for the address generation unit and may reduce the efficiency of the PEs. Additionally, a reduction in the PSNR of the resultant motion compensated residual is to be expected as with any fast search strategy. Swamy et al implemented a hardware architecture for 1D hierarchical search strategy [228]. This architecture is approximately one fifth the size of a conventional 2-D systolic array but achieves marginally faster processing for a range of sequences. Nakayama et al use a fast scene adaptive search strategy along with adaptive thresholds and pixel quantisation to the upper four bits of each pixel [229]. They claim a processing time and power reduction of approximately 90% relative to a conventional 1-D systolic array approach. Huang et al proposed a parallel global elimination architecture, having comparable processing time to a 2-D systolic array but an area of less than a 1-D systolic array architecture [230].

Although general purpose processors have vastly increased computational throughput since the first hardware architectures for motion estimation were proposed, advances in the underlying motion estimation algorithms to reduce the energy in the motion compensated prediction residual have increased the overall computational cost. Techniques such as unrestricted motion estimation, variable block size motion estimation, subpel-motion estimation and multiple reference frames have all contributed to this higher computational cost. This means that dedicated hardware architectures for motion estimation remain a highly active research area. Recent research has extended the classical 1-D and 2-D systolic array architectures to handle variable block sizes. Yap and McCanny modified the classical 1-D systolic array full search architecture to generate all 41 combinations of motion vectors in a H.264 16×16 macroblock [231]. In the architecture the SAD value for each 4×4 subblock is combined in a merging circuit to generate SAD values for all possible block configurations. Variants of the 2-D systolic array architecture were also proposed for H.264 by many including Kao et al and Deng et al [232][233]. Architectures for variable block size fast search strategies have also been proposed. Wei et al presented a number of architectures capable of handling three step search, four step search, logarithmic 2-D search and the full search strategy [234][235][236]. To reduce the energy in the prediction residual Rahman and Badawy proposed a quarter-pel full search systolic array architecture [33]. This uses eight processing units, where each processing unit is 1-D 16 PE systolic array.

An another important consideration for dedicated hardware architectures is the vast quantity of pixels required by the block matching algorithm, as such memory architecture is commonly investigated for hardware implementations. Lai and Chen proposed a data interlacing technique to exploit overlapped search area for a 2-D systolic array architecture [237]. The approach reduces

the number of external memory access as well as potentially improving the PE efficiency. Tuan et al also examined memory issues, analysing the redundancies inherent in different full search systolic array architectures caused by retrieving the same pixel multiple times [238]. Local memory organisation within the context of H.264 was addressed by Song et al [239].

With the emergence of mobile multimedia, energy efficiency has come to the forefront of design space considerations for development of battery powered mobile device. As motion estimation is the most computationally demanding function in a modern video codec, it is no surprise that low power architectures for motion estimation have been proposed to address the need for improved energy efficiency. At a high level of abstraction, power minimisation techniques for the motion estimation block matching algorithm can be considered to reduce switching activity in the search strategy and/or the block matching. Reducing the number of search points in the search strategy is a common technique employed in software implementations to improve throughout, but it could also be considered to reduce power consumption. The disadvantage of this approach is that it is widely acknowledged to reduce the quality of the prediction residual[1] [32]. This means reduced video quality for a given fixed bitrate or a higher bit rate for similar quality. A higher bitrate will create additional switching activity throughout the codec and in a mobile video telephony application scenario will cause extra switching in the wireless RF circuitry on the mobile device. An alternative approach, that is some times known as a fast exhaustive search or SAD summation truncation [32], uses all positions defined in the full search strategy, but where possible terminates the calculation of block matches early, thus saving processing cycles and power consumption. This early termination condition is if the partial SAD[2] accumulated for the block match is larger than the minimum SAD found so far within the search window. The reason it is possible to terminate processing early with this condition is that further processing will only make the SAD result larger and thus the final SAD result will also be greater than the minimum SAD. A result larger than the minimum SAD is of no use since the motion vectors will be chosen on the basis of the block with a minimum SAD value.

Early termination of block matching is frequently used in software implementations and to date in a relatively small number of hardware implementations [240][241][242][243][244]. In dedicated hardware, Do and Yun highlighted that provided that the cost of computing whether the minimum SAD has been exceeded is negligible compared to the full SAD calculation, consid-

[1] A motion compensated prediction residual found using a reduced number of search positions has a high probability of having greater energy than if the prediction residual was found using a full search strategy

[2] The techniques applies equally well to other distortion metrics including mean absolute difference, mean square error etc.

erable power savings are possible. Do and Yun proposed a modification to a 2-D systolic array architecture that uses a preliminary conservative SAD estimate prior to the block match. This conservative SAD estimate is always less than the true SAD value, which means if the conservative SAD estimate is greater than the minimum SAD found so far, computations for the remainder of the block match can be halted. Due to the systolic array architecture, it is not viable to skip to the next block match and reduce the number of processing clock cycles, as doing so would cause excessive disruption to the data flow within the systolic array. Nevertheless by gating off the absolute difference when the minimum SAD is exceeded Do and Yun reported power savings of approximately 50%. Lam and Tsui also proposed a 2-D systolic array with early termination of the SAD processing, though the granularity of termination is not as fine, so the reported power savings are less at 30% to 40% [242]. Sousa and Roma proposed a low power linear systolic array architecture, in which the processing is disabled (via blocking registers) if the currently calculated SAD exceeds the minimum SAD [241]. The check for the disabling condition is carried out at the end of each line. As this architecture is a linear systolic array, the required clock frequency is obviously higher than that required by the 2-D systolic array as proposed by Do and Yun, but the silicon area is also smaller. Sousa and Roma also report power saving of approximately 50%. Takahashi et al proposed a 1-D systolic array with eight PEs in conjunction with early SAD termination [244]. When all eight PEs terminate processing early, the systolic array is cleared and new block matches commence. This architecture is reported to make 45% to 51% savings in power consumption. Richmond and Ha uses the early termination technique with a fast four step search strategy [243]. Similar to other architectures discussed, if the minimum SAD is exceeded during a block match the inputs to the PE are gated off. The early termination coupled with the fast search resulted in power saving of 75% to 85% compared to a conventional full search (though it is not clear whether this related to a 1-D or 2-D systolic array) . However, due to the fast search there was an associated reduction of the PSNR of the motion compensated residual by over 0.5dB in high motion sequences. Lopez et al adapted an early SAD termination 1-D systolic array architecture for H.264 variable block motion estimation [245]. However, the scope for early termination is reduced due to the need for different configurations of block sizes. The silicon resources also increased by a factor of 2.5 and the maximum clock frequency reduced by approximately 30%.

7.1.1 Binary Motion Estimation

Another method of reducing the computational cost associated with the block matching algorithm is to minimise the complexity of the matching criteria using pixel quantisation techniques. Many of these techniques manipulate the pixels into a binary form so that simple XOR logic can replace the absolute difference between pixels. Whilst this may initially only seem like a slight modification, because the matching criteria is the inner most loop of deeply nested loops in the block matching algorithm, such a modification can dramatically reduce the computational cost. As such many different approaches for binary block matching have been proposed. This prior research can be broadly categorised as either bit-plane matching based motion estimation or the more popular fully binary quantised pixel based motion estimation.

Ko et al proposed a bit plane matching approach, in which each bit of each pixel is treated independently from the other bits of the pixel [246]. In this way the absolute difference can be calculated using only an XOR between the bits of each bit plane. To improve the performance Ko et al use a correlation index to combine the results from each bit plane. Ko et al also suggested a 2D systolic array hardware architecture to implement this scheme. Celebi et al also proposed a hardware architecture for bit plane matching [247]. In addition, Celebi et al add redundancy to each pixel via a pre-coding stage in order to improve the quality of the motion vectors.

An alternative approach to bit-plane matching, which can reduce the computational cost even further, is to fully quantise each pixel to a binary value prior to motion estimation. The block matching then uses a single bit XOR to evaluate the absolute difference between pixels. This method does requires a pre-processing filter to reduce the eight bit pixel value to a single bit binary representation. Nevertheless the savings can be substantial, it was observed by Mizuki et al, that even after including the binary filtering-stage, the overall silicon area was reduced by a factor of more than five compared to motion estimation using the same full search strategy with 8-bit resolution pixels [248]. Mizuki et al used an edge filter to generate the binary pixels. Natarajan proposed a 1-D systolic array architecture using binary quantised pixels [249]. The architecture uses 16 PEs, with each PE operating on 16 1-bit pixels. Once the pixel values are XOR'ed, a LUT is used to generate the summation. Using this architecture along with a ± 8 search window, a full search strategy and a block size of 16×16, one set of motion vectors is generated every 271 clock cycles. This compares to in excess of 4000 clock cycles for a conventional 8-bit pixel 1-D systolic array architecture such as that proposed by Yang et al [223]. A detailed view of the processing element is shown in Fig. 7.2. Natajaran et al generated the binary pixels via the convolution of a

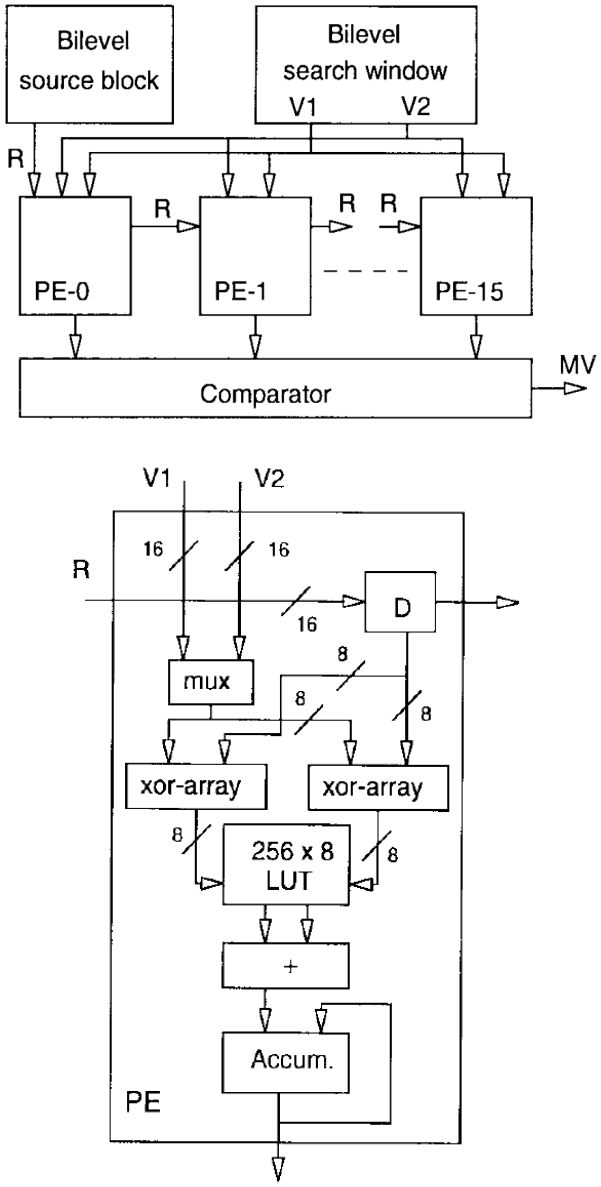


Figure 7.2: Systolic array architecture and PE proposed by Natajaran et al [249]

17×17 multi-bandpass kernel with the original pixel data.

Binary motion estimation schemes can be used with any search strategy. In a software implementation by Feng et al, initial searching is seeded using a local context of previously generated motion vectors [250]. The local context of motion vectors are also compared to threshold values to dynamically alter the search window size. This adaptiveness is used in the binary motion estimation phase, which discounts bad matches prior to an 8-bit resolution full search strategy. In another software implementation, Wong et al couples binary motion estimation with an adaptive two step search strategy, and an optional full search refinement stage [251]. The binary filtering used by Wong et al is the same as that proposed by Natajaran et al. Song et al proposed a hierarchical

binary motion estimation scheme [252]. In this scheme the pixels in the lowest layer (i.e. smallest resolution) in the hierarchical pyramid are in 8-bit format, whilst the pixels in the remaining layers are in binary format. Luo et al also proposed a hierarchical binary motion estimation algorithm, but removed the requirement for the pixels in the lowest layer to have 8-bit resolution [253]. A full search with a ± 3 search window is firstly conducted at the lowest layer. The best candidate from this search is compared to five other spatio-temporal candidates. The best match from these six candidates is used to seed the starting location at the next layer. At the third and final layer a refinement search with ± 2 search window is conducted around the location of the best match from the second layer. .

In prior research that applies binary motion estimation to 8-bit texture pixels, much of the innovation occurs in the quantisation of the pixels to the binary value, as this impacts the quality of the motion compensated result. For example, a good binary quantisation filter accurately captures both local detail and edge information, allowing the binary motion estimation phase to find a motion vector which will give a close approximation to the optimum motion vectors found using full resolution pixels. Within the context of binary motion estimation for MPEG-4 binary shape coding, the binary quantisation of pixels is not necessary as the alpha pixels are already binary in nature. Binary motion estimation research for MPEG-4 shape coding has principally focused on fast heuristic search strategies. Yu et al. outline a software implementation for motion estimation for shape, which uses a number of intermediate thresholds in a heuristic search strategy to reduce the computational complexity [254]. The author does not consider this approach viable for a hardware implementation, owing to the irregular memory addressing, in addition to providing limited scope for exploiting parallelism. Using a shape boundary mask can also be employed in a preprocessing manner to reduce the number of search positions [255][256]. However the generation of the boundary mask by Panusopone et al. may be considered to be computational intensive owing to the block loop process [255]. Tsai et al. use a more efficient approach for generating the boundary mask [256]. Furthermore, Tsai et al. use heuristics to further reduce the search positions and additionally discuss a possible hardware architecture. Chang et al. use a 1D systolic array architecture coupled with a full search strategy for the Binary Motion Estimation (BME) implementation [257]. Improving memory access performance is a common optimisation in MPEG-4 binary shape encoders [258][259]. Lee et al. suggest a run length coding scheme to minimise on-chip data transfer and reduce memory requirements, however the run length codes still need to be decoded prior to BME [259]. The proposed solution, which is described in the

subsequent section, leverages early SAD operation termination and attempts to avoid unnecessary operations by exploiting redundancies in the binary shape information. This is in contrast to a systolic array approach, where unnecessary calculations are unavoidable due to the data flow in the systolic structure. Unlike the approach of Tsai et al., the proposed approach uses an exhaustive search to guarantee finding the best block match within the search range [256].

7.2 Proposed Binary Motion Estimation Architecture

In this thesis, motion estimation is considered to follow object segmentation. It follows then that the input pixels to the motion estimation block are in binary form. With this assumption this research attempts to reduce the power consumption of motion estimation using two techniques which will not jeopardise the quality of the prediction residual. The first technique employed is early SAD termination, which was described in Section 7.1. This uses all positions defined in the full search strategy, but terminates processing when the minimum SAD is exceeded, thus saving processing cycles and power consumption. The challenge with implementing early termination in conventional systolic array hardware architectures is that to attempt to finish processing early for one block match in one PE and subsequently load the next block match would cause too much disruption to the overall data flow. Whilst the author accepts that systolic arrays offer many attractive features, this research explores whether alternative architectures would make it possible to save power consumption elsewhere. In the proposed architecture (which is not systolic array based), to exploit the potential of early termination during each block match the partial SAD calculated to date is subtracted from a register, which is initially loaded with the minimum SAD value calculated at that point. If a sign change occurs during the de-accumulation step, there is no need to continue further processing since the current minimum SAD has already been exceeded. This method reduces the number of calculations to find the motion vectors by discounting bad matches early on, and thus power is saved. This process is inherently serial, so to improve throughput, pixels in the block match are repartitioned to allow parallelism to be exploited. Furthermore, the pixels are repartitioned in such a manner so as to increase the probability that all parallel PEs will terminate at the same point. The features used to achieve this in the proposed architecture are described in detail later in Section 7.2.2.

In previous binary motion estimation research, no attempts have been made to optimise the SAD PE datapath. However, the unique characteristics of the SAD calculation for binary data

mean further redundancies can be exploited to reduce datapath switching activity. From Eqn. 7.1 it is clear that there are unnecessary memory accesses and operations when both B_{cur} and B_{ref} pixels have the same value, since the XOR will give a zero result. As such the second algorithmic-level power minimisation technique proposed in this thesis is based on exploiting these redundancies. To achieve this, a scheme is proposed which reformulates the conventional binary SAD equation into a format which is amenable for a pixel addressing scheme which minimises these redundancies. This is described in detail in the next section.

$$SAD\left(B_{cur}, B_{ref}\right) = \sum_{i=1}^{i=M} \sum_{j=1}^{j=N} B_{cur}\left(i,j\right) \otimes B_{ref}\left(i,j\right) \tag{7.1}$$

7.2.1 Reformulation of the Binary SAD Metric

The conventional equation for calculating the binary SAD is given in Eqn.7.1. The first step in reformulating this equation is to observe the following properties from Fig. 7.3:

$$TOTAL_{cur\_blk} = COMMON + UNIQUE_{cur\_blk} \tag{7.2}$$

$$TOTAL_{ref\_blk} = COMMON + UNIQUE_{ref\_blk} \tag{7.3}$$

where

- $TOTAL_{cur\_blk}$ is the total number of white pixels in the current macroblock
- $TOTAL_{ref\_blk}$ is the total number of white pixels in the reference macroblock.
- $COMMON$ is the number of white pixels that are common in both the reference macroblock and the current macroblock
- $UNIQUE_{cur\_blk}$ is the number of white pixels in the current macroblock but not in the reference macroblock.
- $UNIQUE_{ref\_blk}$ is the number of white pixels in the reference block but not in the current macroblock.

It is also clear from Fig. 7.3, that because the pixels are in binary form, the SAD value between the current and reference macroblock can be represented as:

$$SAD = UNIQUE_{cur\_blk} + UNIQUE_{ref\_blk} \tag{7.4}$$

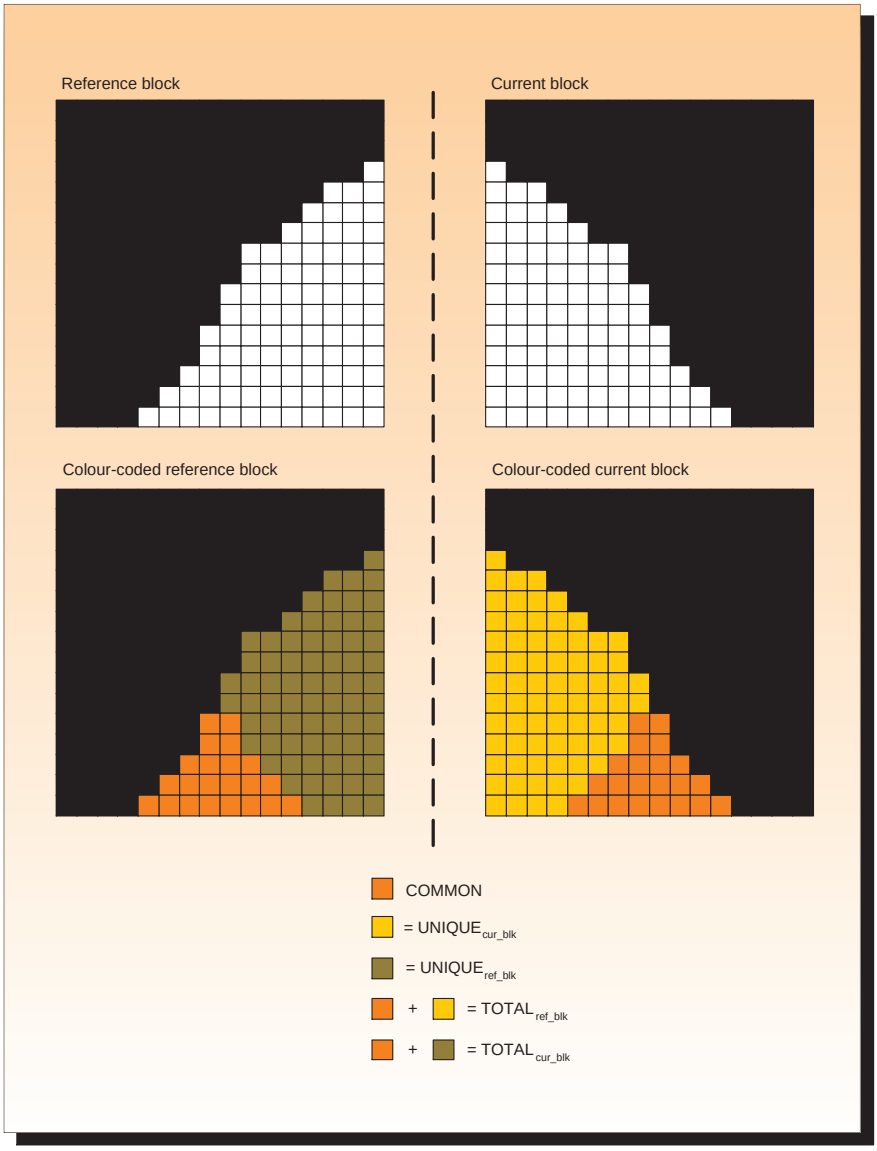


Figure 7.3: SAD Reformulation

Using the identifies in Eqn. 7.2 and Eqn. 7.3, it follows that:

$$UNIQUE_{ref\_blk} = TOTAL_{ref\_blk} - (TOTAL_{cur\_blk} - UNIQUE_{cur\_blk}) \quad (7.5)$$

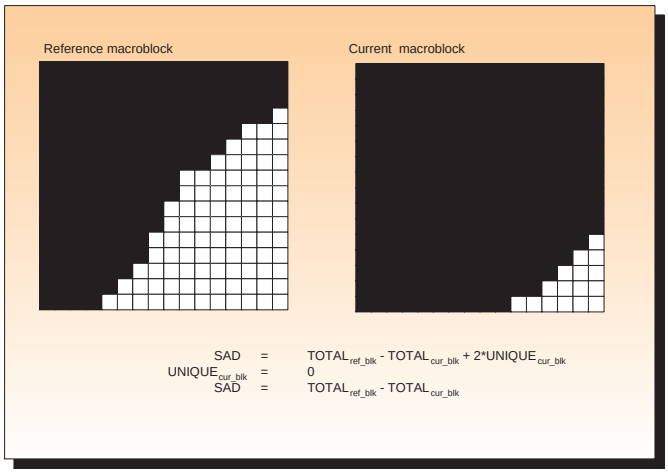
Substituting Equation 7.5 into Equation 7.4, gives the following:

$$SAD = TOTAL_{ref\_blk} - TOTAL_{cur\_blk} + 2 \times UNIQUE_{cur\_blk} \quad (7.6)$$

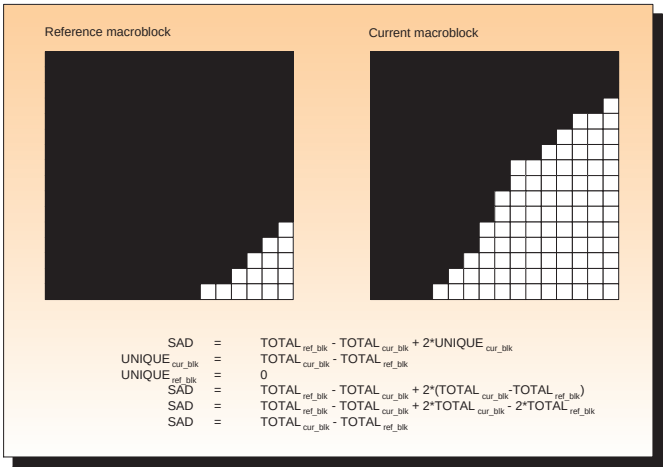
To intuitively understand Eqn. 7.6, $TOTAL_{ref\_blk} - TOTAL_{cur\_blk}$ can be considered to be a conservative estimate of the SAD value, whilst $2 \times UNIQUE_{cur\_blk}$ is an adjustment to the conservative SAD estimate to give the correct final SAD result. To aid clarity three different macroblock scenarios are shown in Fig 7.4, in each case the calculation of the SAD value is generated using Eqn. 7.6. The reason Eqn. 7.6 is beneficial is because:

- $TOTAL_{cur\_blk}$ is calculated only once per search window
- It is possible to update $TOTAL_{ref\_blk}$ in 1 clock cycle (see Section 7.2.1.1 for more details).
- If $TOTAL_{cur\_blk} = 0$, the SAD value is $TOTAL_{ref\_blk}$ and is calculated in a minimal number of clock cycles.
- Similarly if $TOTAL_{ref\_blk} = 0$, the SAD value is $TOTAL_{cur\_blk}$ and is calculated in a minimal number of clock cycles.
- If $TOTAL_{ref\_blk} = 255$ all the white pixels in the current block will overlap with all the white pixels in the reference block. This means $UNIQUE_{cur\_blk}$ is zero, which allows the SAD to be calculated in a minimal number of clock cycles.
- Incremental addition of $UNIQUE_{cur\_blk}$ allows early termination if the current minimum SAD is exceeded
- The use of $UNIQUE_{cur\_blk}$ eliminates unnecessary memory accesses and operations.

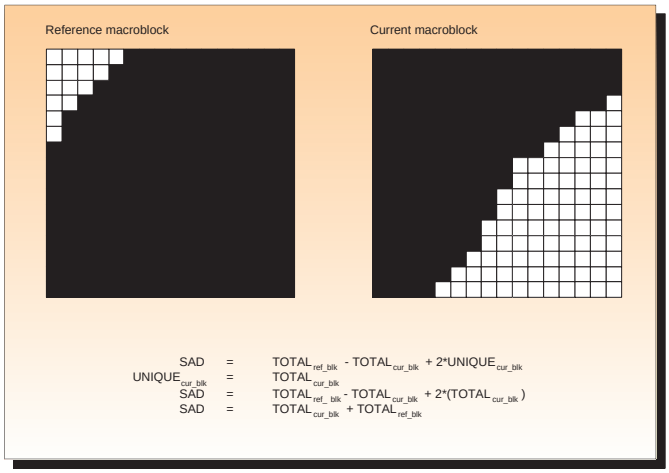
For practical implementation purposes, it is not possible to know $UNIQUE_{cur\_blk}$ in Eqn. 7.6 in advance of a block match. By definition $UNIQUE_{cur\_blk}$ is the summation of the white pixels in the current block where the collocated pixel in the reference block is a black pixel. Fortunately, it is possible to process only the white pixels in the current macroblock, which are then incrementally XORed with the pixel in the collocated position in the reference macroblock which then finally



(a) Example 1



(b) Example 2



(c) Example 3

Figure 7.4: Reformulated binary SAD Examples

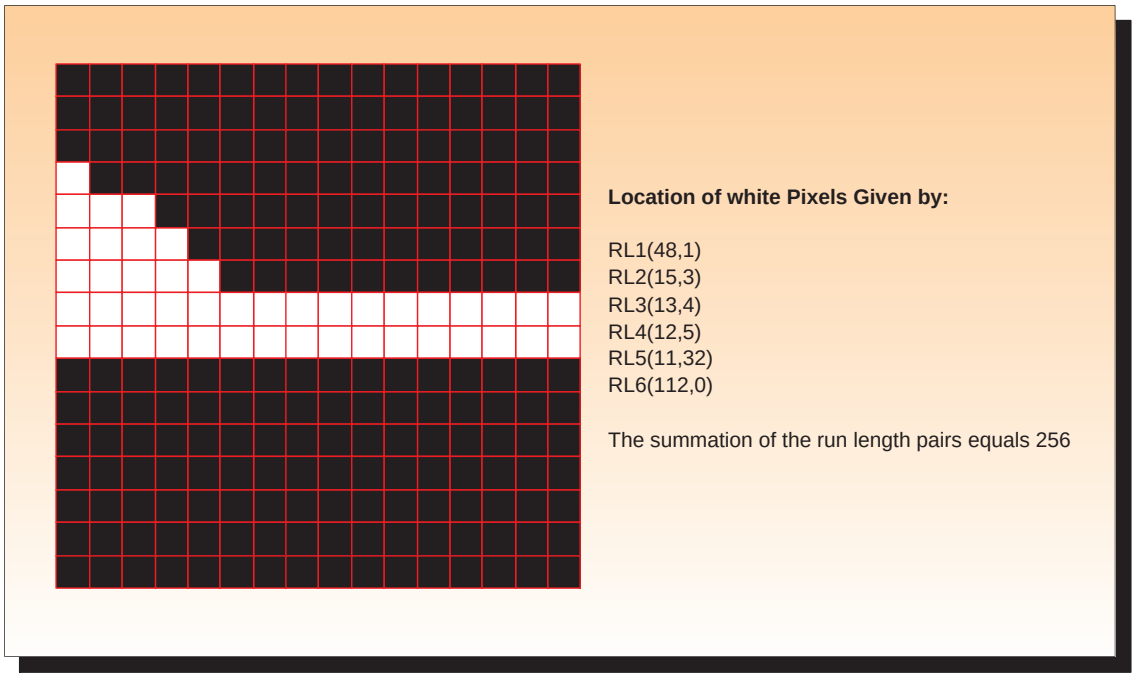


Figure 7.5: Example of Run Length Coding

give $UNIQUE_{cur\_blk}$. This minimises[3] access to irrelevant data and reduces switching activity. To access only the white pixels in the current macroblock, the author proposes that the pixels undergo *run length coding*. Each run length code consists of two elements, an *offset* and a *run*. The offset represents the number of pixels until the next pixel of interest (which in this case is white) is encountered. Whilst the run element is the number of consecutive pixels of interest. A further point worth noting is that in a $N \times M$ macroblock, the summation of the offset and run components equals $N \times M$. As will be described in Section 7.2.1.1, this property is exploited in generating an alternative mode for the calculation of the binary SAD. An example run length encoding of a arbitrary macroblock is shown in Figure 7.5. In this case the first run length code for is: $RL1$ = (48,1), meaning there are 48 black pixels before a white pixel is encountered. The value of one means that there is only one white pixel in this "group". In the proposed architecture the run length codes are generated in parallel with the first match of the search step. It is possible to do this because no minimum SAD exists for the initial block, causing an arbitrarily large value to be loaded into the SAD de-accumulation register, which makes early termination of SAD processing impossible. As the first match always takes $N \times N$ (where N is the block size) cycles to complete, this provides amble time for the run length encoding process to operate in parallel. After the RLC encoding, the logic can be powered down or clock gated until the next current block is processed.

[3]If the reference pixel is white a redundant operation will still occur

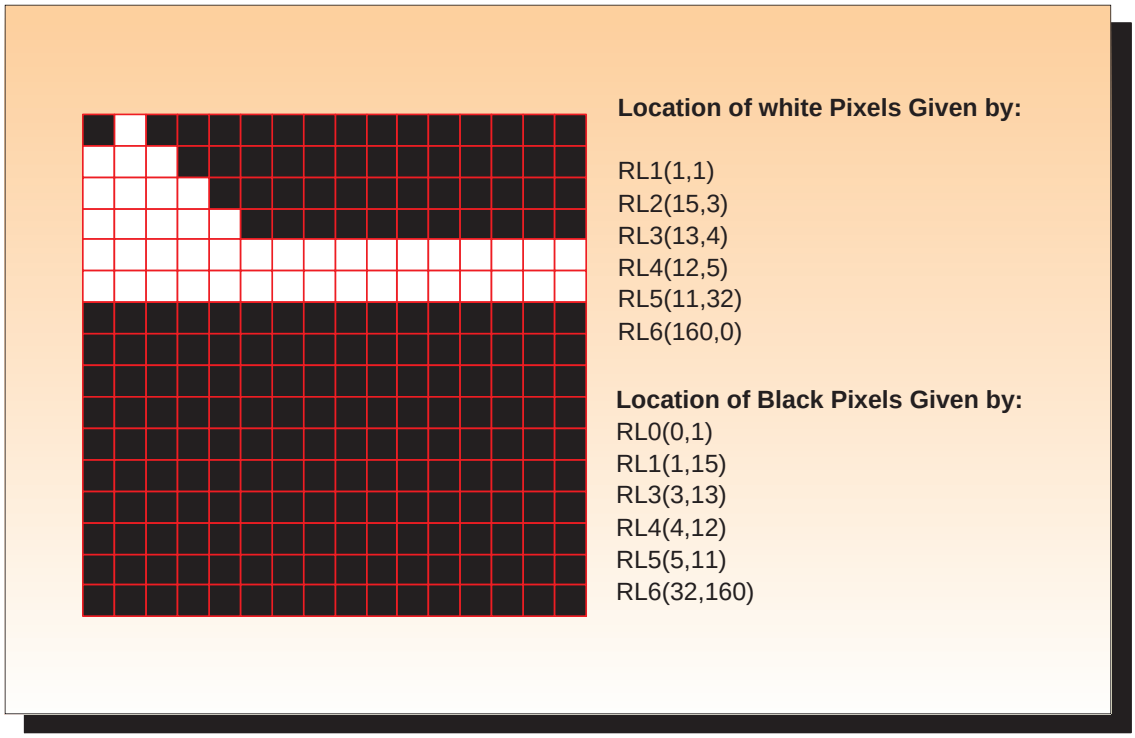


Figure 7.6: Regular and Inverse RLC pixel addressing

7.2.1.1 Using inverse run length codes

The discussion involving Eqn. 7.6 has so far focused on the incremental addition of the $UNIQUE_{cur\_blk}$ SAD adjustment factor. However, by using a different combination of Eqn 7.2, Eqn. 7.3 & Eqn 7.4, the SAD calculation can be expressed in terms of $UNIQUE_{ref\_blk}$.

$$SAD = TOTAL_{cur\_blk} - TOTAL_{ref\_blk} + 2 \times UNIQUE_{ref\_blk} \tag{7.7}$$

This is shown in Eqn. 7.7. The reason this is beneficial will be explained shortly, after firstly examining the properties of $UNIQUE_{ref\_blk}$. To generate $UNIQUE_{ref\_blk}$ it is not practical to run length encode the reference macroblock in a similar manner to the way it was carried out for the white pixels in the current macroblock in Eqn. 7.6. This is because the run length code would change for each new search position. For example in a linear search when a new reference block is selected, the new position is offset by one row/column from the previous search position. Since a run length code is always relative to a fixed position (i.e. the first pixel in the block), the run length code for each new reference block would need to be regenerated. This is not computationally attractive for every search position. However $UNIQUE_{ref}$ can also be generated by using the black pixels in the current macroblock, since a unique white pixels in the reference block can only

be "unique" if the collocated pixel in the current block is black. The location of the black pixels can be much more easily derived, by manipulating the run length codes for the white pixels. In general[4] it can be seen in Fig. 7.6 that the "inverse" run length pairs have the pattern of being offset by one component from the regular run length pairs. For example, the second inverse run length pair is $IRL1 = (1, 15)$, the third pair is $IRL2 = (3, 13)$ and so on. By reusing the run length code associated with the white pixels, additional memory is not required and furthermore the same SAD datapath can be reused with minimal additional logic.

The motivation for using Eqn. 7.7 instead of Eqn. 7.6 is that under certain conditions it can reduce the number of operations and memory accesses. This can be understood by considering that in the worst case scenario Eqn. 7.6 requires a maximum of $TOTAL_{cur\_blk}$ incremental additions of pixels to generate a final $UNIQUE_{cur\_blk}$. Whereas using Eqn. 7.7 the maximum number of operations is $N \times N - TOTAL_{cur\_blk}$. Therefore calculating the macroblock SAD value using Eqn. 7.7 rather than Eqn. 7.6 is useful for macroblocks in which there are fewer black pixels than white pixels. This means the selection of either SAD equation can be easily chosen based on the most significant bit of $TOTAL_{cur\_blk}$. This also means there is minimal modification to the hardware architecture since the mode can be supported using only one extra multiplexer.

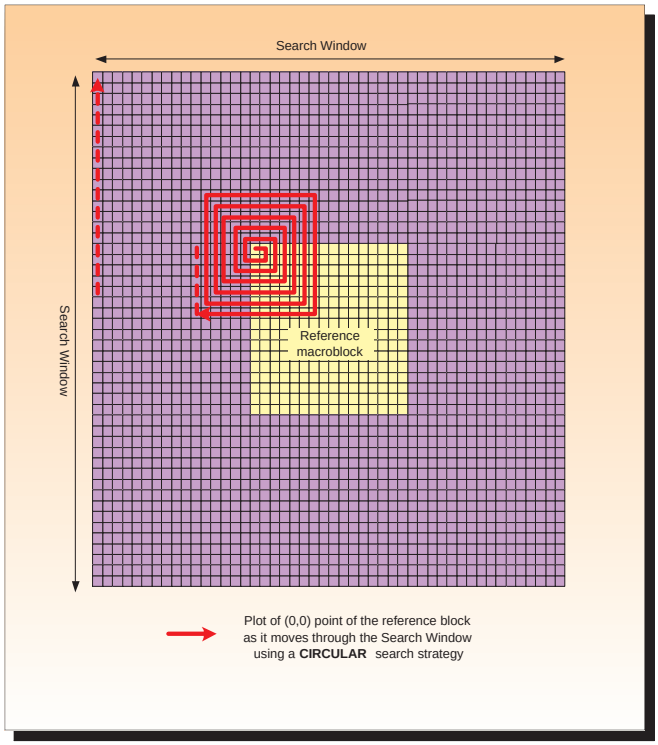
7.2.1.2 $TOTAL_{ref\_blk}$ Update

Similar to $TOTAL_{cur\_blk}$, the initial $TOTAL_{ref\_blk}$ calculation is carried out during the first block match when no early termination is possible. When a block match SAD is fully calculated or terminated early, the address generation unit moves the reference block to a new position. Provided a linear search (such as the circular search or the full search strategies that are shown in fig. 7.7) is used, $TOTAL_{ref\_blk}$ can be updated in one clock cycle. This is done by subtracting the previous row or column (depending on search window movement) from $TOTAL_{ref}$ and adding the new row or column (see Fig. 7.8). A simple adder tree (such as that shown in Fig. 7.8) implements the update within one clock cycle. Alternatively, a carry save structure could also be used.

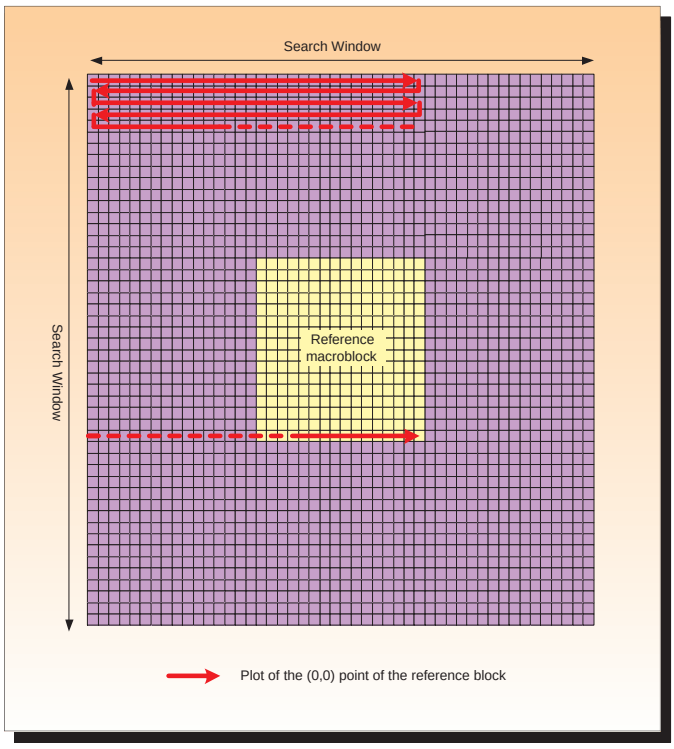
7.2.2 4xPE Block Matching Architecture

The proposed architecture consists of two stages, a block matching stage and a minimum SAD update stage, both of which are run in parallel. The update stage is by default idle and can be

[4]The exception is the first inverse run length pair which requires minimal additional processing



(a) Circular Search



(b) Modified Full Search

Figure 7.7: Search strategies

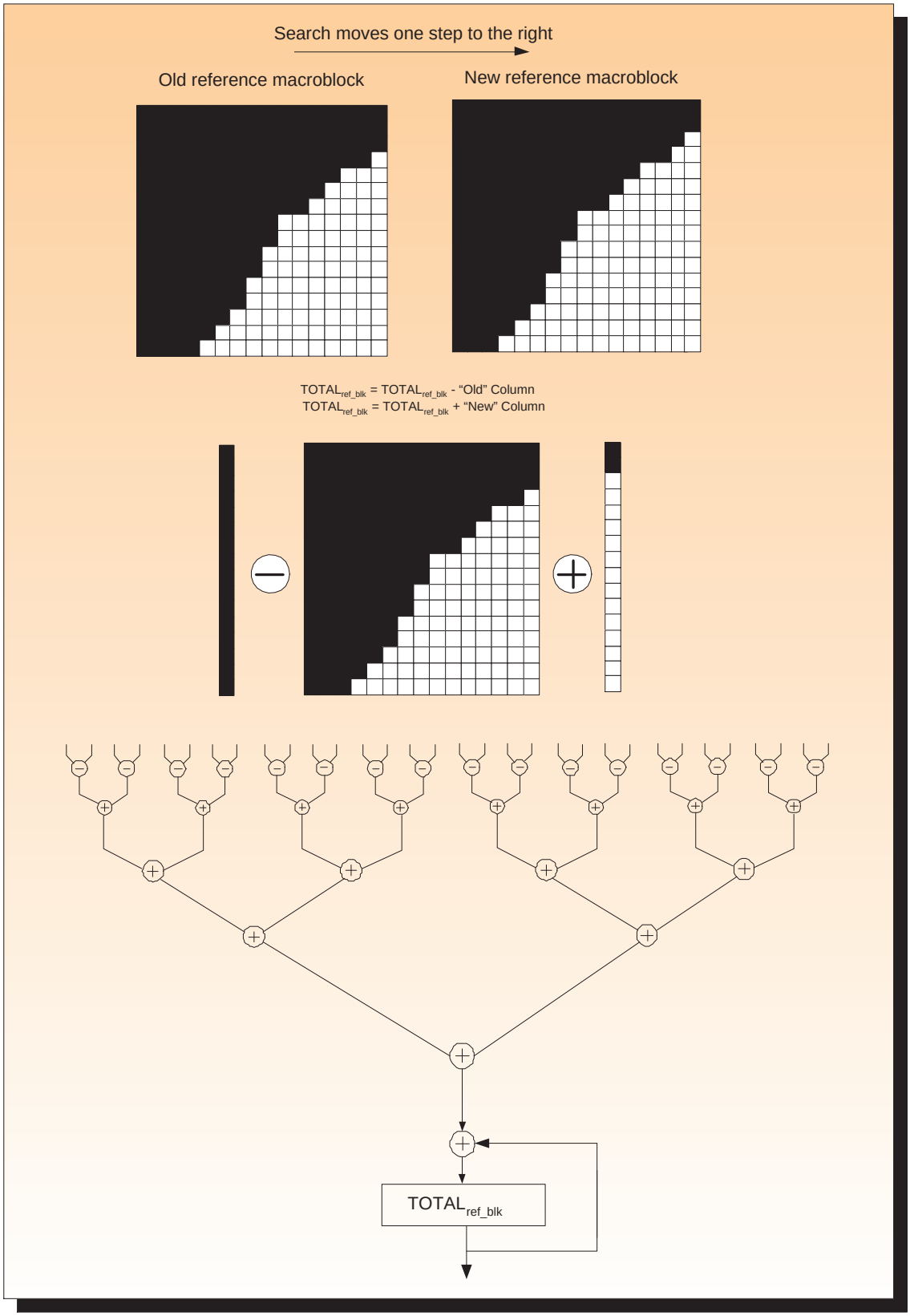


Figure 7.8: $TOTAL_{ref\_blk}$ Update

either powered down or clock gated. In the block matching in order to exploit early termination in the SAD calculation, an intermediate partial SAD must be generated. To gain the greatest benefit from early SAD termination requires the SAD calculation to proceed in a serial manner. However, such serial processing reduces encoding throughput, which is not desirable for real time applications. To increase throughput, parallelism must be exploited. To achieve this, the proposed architecture repartitions each 16×16 macroblock into four smaller 8×8 blocks. A dedicated PE then calculates the SAD for each 8×8 block. This implies a non-systolic array type architecture, since data cannot be shared between PEs. With conventional 8-bit pixel motion estimation this would most likely lead to excessively high memory bandwidth. However due to the nature of the binary data in binary motion estimation for shape coding, there is much higher probability of early termination of SAD processing, which the author proposes will compensate for concerns over memory bandwidth. This will be discussed further in Section 7.3.

At any given clock cycle, the control logic uses the partially accumulated SAD values in each PE to make a decision on whether early termination of the macroblock SAD calculation is possible. The proposed memory repartitioning scheme (used for both the current and reference frames) is shown in Figure 7.9. Rather than use a memory repartitioning scheme that is based on the four quadrants of the macroblock, the proposed scheme results in each PE operating on what appears as a subsampled version of the original macroblock. In this way, if early termination occurs, there is a high probability it should happen at approximately the same time for each PE. The goal is to minimise situations in which one or more PEs have terminated early but must wait considerably longer for the final PE to complete. This is quite likely to occur if memory partitioning based on quadrants is used.

Fig. 7.10 shows a detailed view of the SAD PE. At the first clock cycle the minimum SAD encountered so far for that PE is loaded into the $DACC\_REG$ register. Subsequent processing decrements from this value, with the control logic monitoring the behaviour of the sign bit for indications to allow early termination. At the next clock cycle $TOTAL_{ref\_blk}$ / $TOTAL_{cur\_blk}$ is subtracted from $DACC\_REG$ (depending on $TOTAL_{cur\_blk}[MSB]$ is 0 or 1 respectively). At the subsequent clock cycle $TOTAL_{cur\_blk}$ / $TOTAL_{ref\_blk}$ again depending on whether $TOTAL_{cur\_blk}[MSB]$ is 0 or 1 respectively, is added to $DACC\_REG$. If a sign change occurs at this point the minimum SAD has already been exceeded and no further processing is required. If a sign change has not occurred the address generation unit retrieves the next run length code from memory. If $TOTAL_{cur\_blk}[msb] = 0$ the run length pair code is processed unmodified. On the other hand

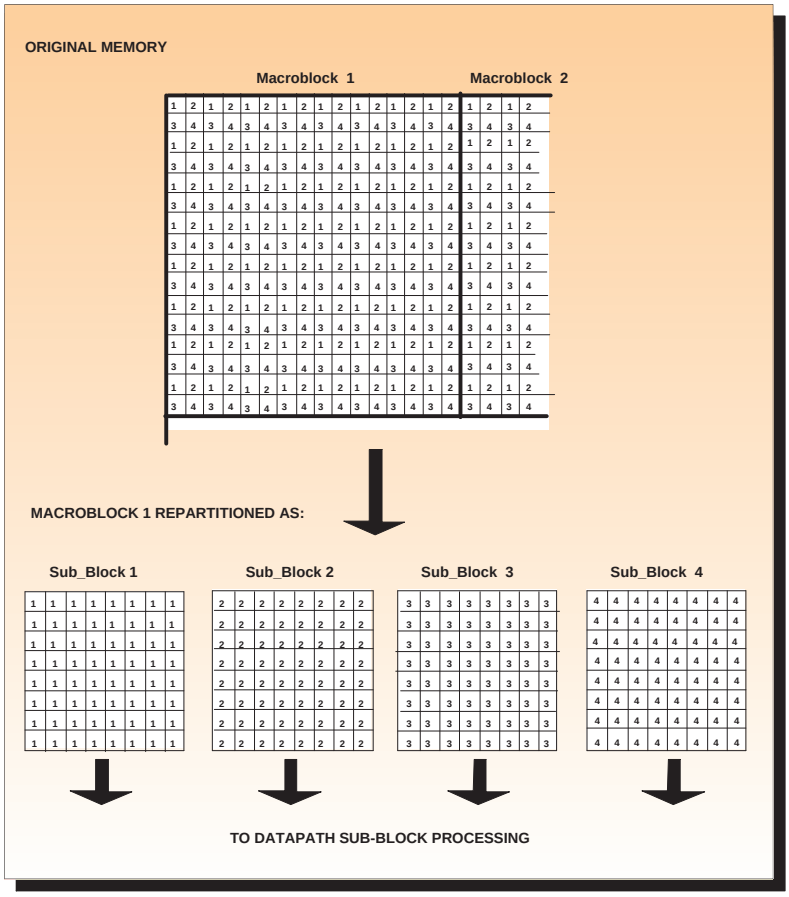


Figure 7.9: Macroblock repartitioning for 4xPE Architecture

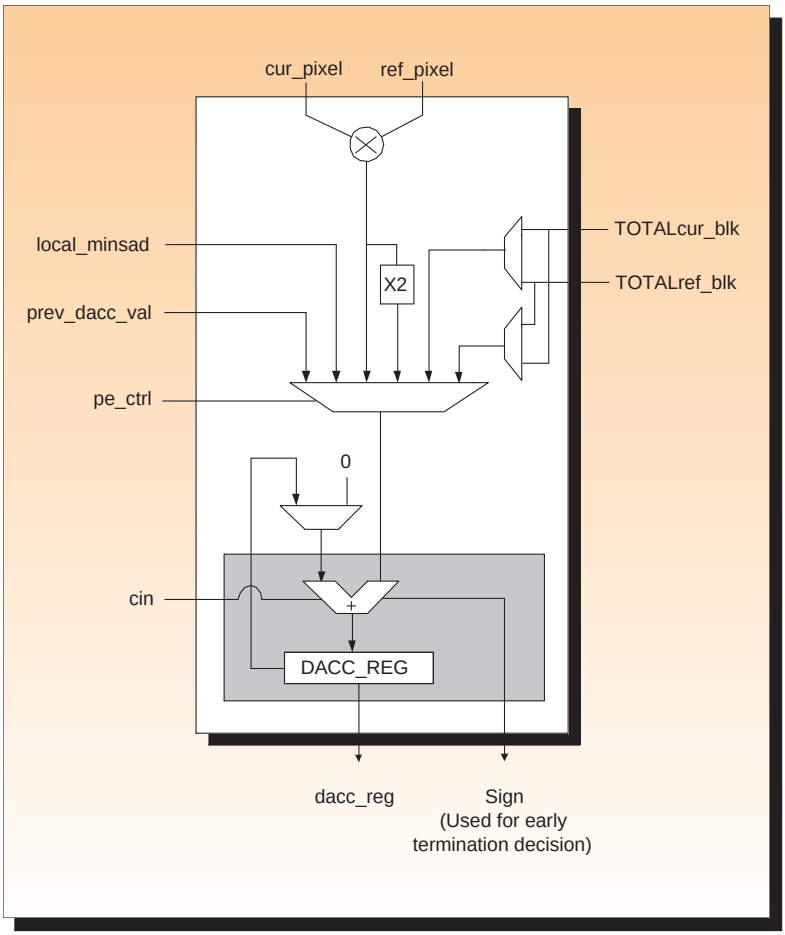


Figure 7.10: RLC SAD Processing Element

if $TOTAL_{cur\_blk}[msb] = 1$ the inverse run length code is processed. In either case the run length code is decoded to give a horizontal and vertical macroblock address. This address is used to retrieve the relevant pixel from the reference macroblock and the current macroblock. The pixel values are XORed and the result is left shifted by one place and then subtracted from the $DACC\_REG$. If a sign change occurs, early termination is possible. If a sign changes does not occur, the remaining pixels in the current run length code are processed. If an early SAD termination signal is still not generated, subsequent run length codes are fetched from memory and the processing repeats. As the block matching progresses, early termination of processing can occur at any point. The early termination can be configured to operate in two different modes. In the default mode of exhaustive search, all four PEs must produce an early termination signal before the processing in this block match is terminated. In fast mode not all PEs need to generate an early termination signal before processing finishes for the current block match. This is because the control logic assumes that the subsampling process will lead to very similar images in each PE and so early termination signals will occur shortly afterwards in each of the remaining PEs. This is especially useful if extra low power operation is required and the data displays a high degree of correlation (i.e. low motion). Although it risks reducing the quality of the motion vectors for scenes with high motion characteristics.

If the block matching progresses until the point where all pixels in the block are processed, the update stage is evoked. This checks if the total SAD from the four PEs is less than the total minimum SAD value encountered from previous block matches. If this is the case a new block level minimum SAD has been found and the current value is updated. The update logic consists of a single adder/subtracter and an accumulator (see Fig. 7.11). Since the PE calculation can take up to $TOTAL_{cur\_blk}$/Inverse $TOTAL_{cur\_blk}$ + 2 steps to complete, it is possible to run the update stage in parallel with a new block match. The update is handled in sequential manner in order to reduce the hardware area overhead. The update takes at most 11 clock cycles to complete. Further SAD cancellations can occur in the next match without affecting the performance of the update logic. If the new block match is cancelled then it would also cancel for the new updated minimum SAD value. This is because the partial SAD value is already bigger than the old minimum SAD so the new minimum SAD will be smaller. The update process firstly accumulates the PE SAD values in the $TOTAL\_DACC\_REG$. If the summation of the PE level SAD values is negative a new minimum SAD has not been found and the update logic can be disabled. Otherwise if the result of this processing is a positive value, a new minimum SAD has been found. The PE level

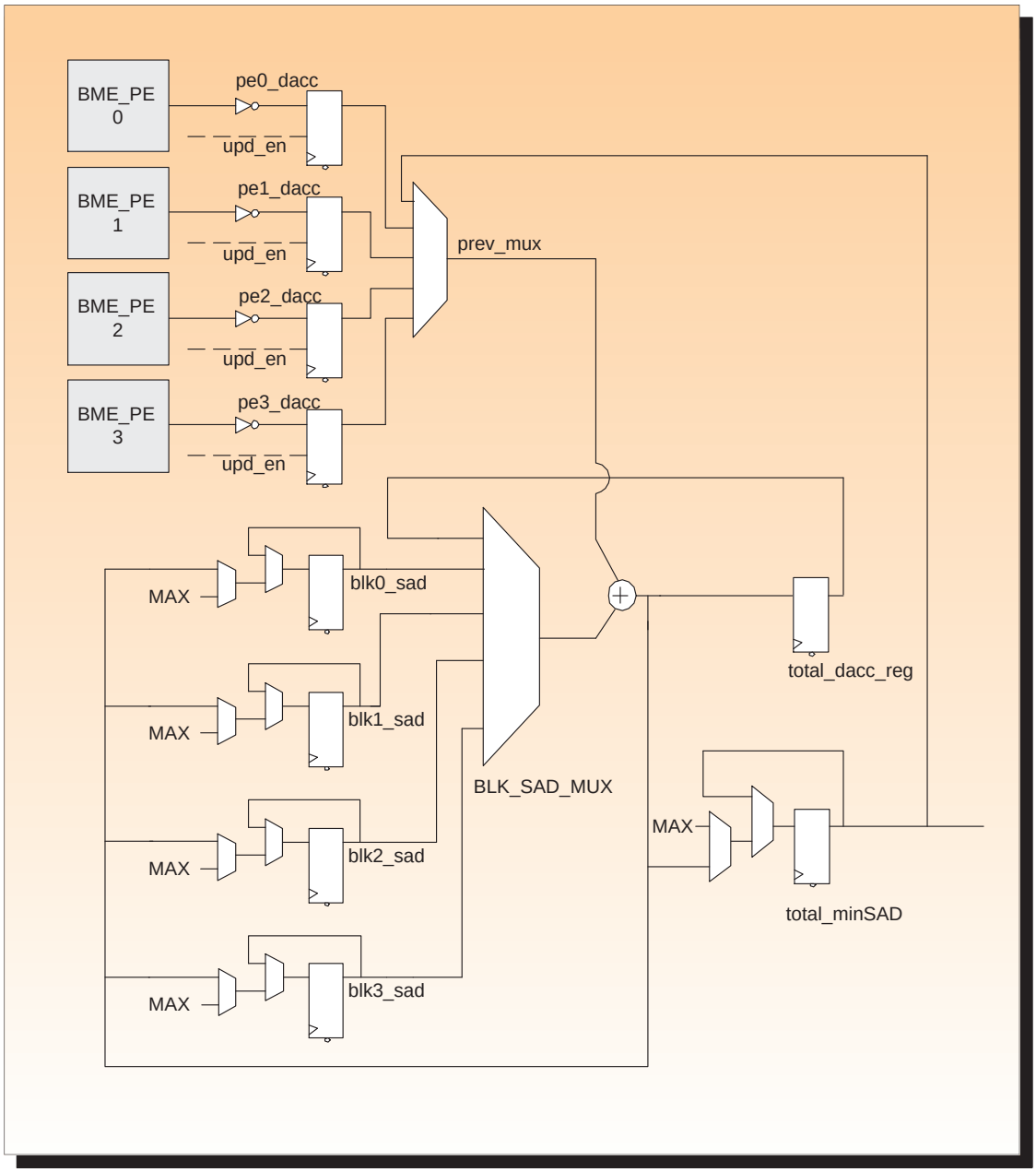


Figure 7.11: 4xPE Update logic

minimum SAD values and total minimum SAD value are then updated. If at any point during an update a SAD cancellation occurs in the new parallel block match, the $DACC\_REG$ will need to be updated with the new minimum PE level SAD value once it is available. Obviously stalling the block match until the update is available is not desirable. In fact it is not necessary to stall the bock match for this lengthy period, instead when the updated value is available each PE is updated with the relative difference between the old and new PE minimum SAD value.

7.3 Results

Unlike the constant throughput systolic array approaches, the processing latency to generate one set of motion vectors for the proposed architecture is data dependant. The worst and best case processing latencies are 65,535 and 3,133 clock cycles respectively. The clock frequency includes a margin to cover below average early termination. As reported in a prior work by the author [260], on average a 93% early termination rate is achieved using common test sequences (see Table 7.1). This figure compares favourable to the early SAD termination hardware designs presented [240][241][244], which typically achieve around 50% reduction in operations. The improvement in the proposed design can be attributed to the subsampling, the use of run length coded addressing scheme and the nature of the binary data for binary shape coding. This early termination figure is used in the subsequent calculation of metrics for benchmarking. It should also be noted that benchmarking is difficult owing to a lack of information in prior art, this includes binary motion estimation architectures used in MPEG-4 binary shape coding and binary motion estimation architectures used in low complexity approaches for texture motion estimation [256][258][257][248][249].

With 93% average cancellation, for a CIF sized video sequence, a total of $65535 \times (\frac{100-93}{100}) \times \left(\frac{352}{16} \times \frac{288}{16}\right) \times 25fps = 45.416 \times 10^6$ clock cycles are required for 1 second of video. Therefore a minimum clock period to achieve this throughput should be 22ns. Table 7.2 summarises the synthesis results for the proposed architecture. Synthesising the design with Synopsys Design Compiler targeting TSMC 0.09μm TCBN90LP technology library yields a gate count of 10,117 and a maximum operating frequency f_{max} of 700MHz. The systolic array binary motion estimation architecture proposed by Natarajan et al., is leveraged in the designs proposed by Chang et al. and Lee et al. Consequently similar cycle counts can be observed in each implementation [249][257][258]. The average cycle counts (4588 cycles) for the proposed architecture is longer

Table 7.1: RLC BME_4xPE versus Conventional Systolic Array BME

	Memory Accesses (1 bit pixels)		Operations (1 bit XOR & addition)	
Sequence	**1-D SA [249]**	**BME_4xPE**	**1-D SA [249]**	**BME_4xPE**
Akiyo	1.5206×10^9	4.5298×10^8	4.0170×10^9	2.6105×10^8
Hall Monitor	1.5206×10^9	4.8202×10^8	4.0170×10^9	2.7847×10^8
Foreman	1.5206×10^9	4.7137×10^8	4.0170×10^9	2.6942×10^8
Average	69.17% reduction		93.29% reduction	

than the architecture proposed by Chang et al and Lee et al [257][258]. This is due to to the architectural design decision to trade off throughput for reduced SAD operations and consequently reduced power consumption. As a consequence of the longer latency, the product of the gate and clock cycles (PGCC, a commonly used metric) for the proposed architecture will be inferior to that of the architecture proposed by Chang et al and Lee et al [257][258]. However, the PGCC metric does not take into account the non-uniform switching in the proposed architecture. For example, after the first block match the run length encoder associated with each PE is not active, in addition the linear pixel addressing for the first block match is replaced by the run length decoded pixel scheme for subsequent block within the search window. The power and energy take account of the non-uniform data-dependant processing, however, benchmarking against prior art using these metrics is not possible owing to a lack of information in the literature. Overall it is reasonable to say that the proposed design is comparable to prior art, trading off throughput for lower power consumption.

Table 7.2: BME Synthesis Results and Benchmarking

Architecture	**Tech** [μm]	**Cycle Count**			**Gates**	**f** [MHz]	**Power** [mW]	**Energy** [nJ]
		Max	**Min**	**Average**				
Mizuki [248]	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a
Natarajan [249]	n/a	1039	1039	1039	n/a	n/a	n/a	n/a
Tsai [256]	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a
Lee [258]	n/a	1056	1056	1056	n/a	n/a	n/a	n/a
Chang [257]	0.35	1039	1039	1039	9666	40	n/a	n/a
Proposed	0.09	65535	3133	4588	10117	100	1.22	80

7.4 Future Work

This section outlines possible avenues for future work. These possible improvements principally focus on reducing the long run time of the proposed architecture and where possible reducing the memory bandwidth.

7.4.1 16XPE Block-Matching Architecture

The architecture proposed in Section 7.2 uses four parallel processing elements. To improve throughput it is possible to use 16 processing elements, although this reduces the granularity of the early termination and as such fewer redundant operations are saved. A further consequence of the 16xPE architecture is that more operations are required for the update stage. An additional 16 BLK_BC_i registers are required to hold the value of the current minimum SAD for each of the 16 PEs. Therefore, whilst the PE datapath structure remains identical, major modifications are required in the update logic for a possible 16xPE architecture. In the 4xPE architecture the update proceeded in a sequential manner requiring maximally 11 cycles. If the same structure was adopted for the 16xPE the minimum update cycle increases to 16 cycles. However the block size has now been reduced to 4×4, meaning that the maximum SAD calculation time is 16 cycles. Therefore the update logic actually runs for as long as the SAD calculation. To prevent excessive stalling in the datapath, the sequential update could be replaced by an adder tree structure. The other major change compared to the 4xPE architecture is that the memory also needs to be remapped from four blocks of 8×8 to sixteen blocks of 4×4. The same pixel subsampling technique is used and the structure can be seen in Fig. 7.12.

7.4.2 Possible PE improvements

While the datapath and update stages of the BME architecture are area efficient, the memory required to store the run length codes and the overhead of generating and decoding the run length codes increase the overall area significantly. If a design is area sensitive, this is far from a desirable situation and in deep sub micron technologies the additional area will only serve to increase static power consumption. The principal reason for the large memory footprint is that worse case scenario's must be handled. For example in the 4xPE architecture each PE processes 8x8 sub block and potentially all 64 pixels could be black or white, thus 7 bits are required to stored this run length. At the other extreme, the sub block could consist of a checker board pattern of alternating single black/white pixels. In this scenario the memory must have storage capacity for 32 run length pairs. As these two situations represent the extreme case a normal sub block is likely to use considerably less memory. There are a number of possible work arounds to reduce the memory requirement. Instead of storing an offset and run length pair, if the colour of the first pixel of the block is stored along with the transitional points the memory space and area can effectively be halved.

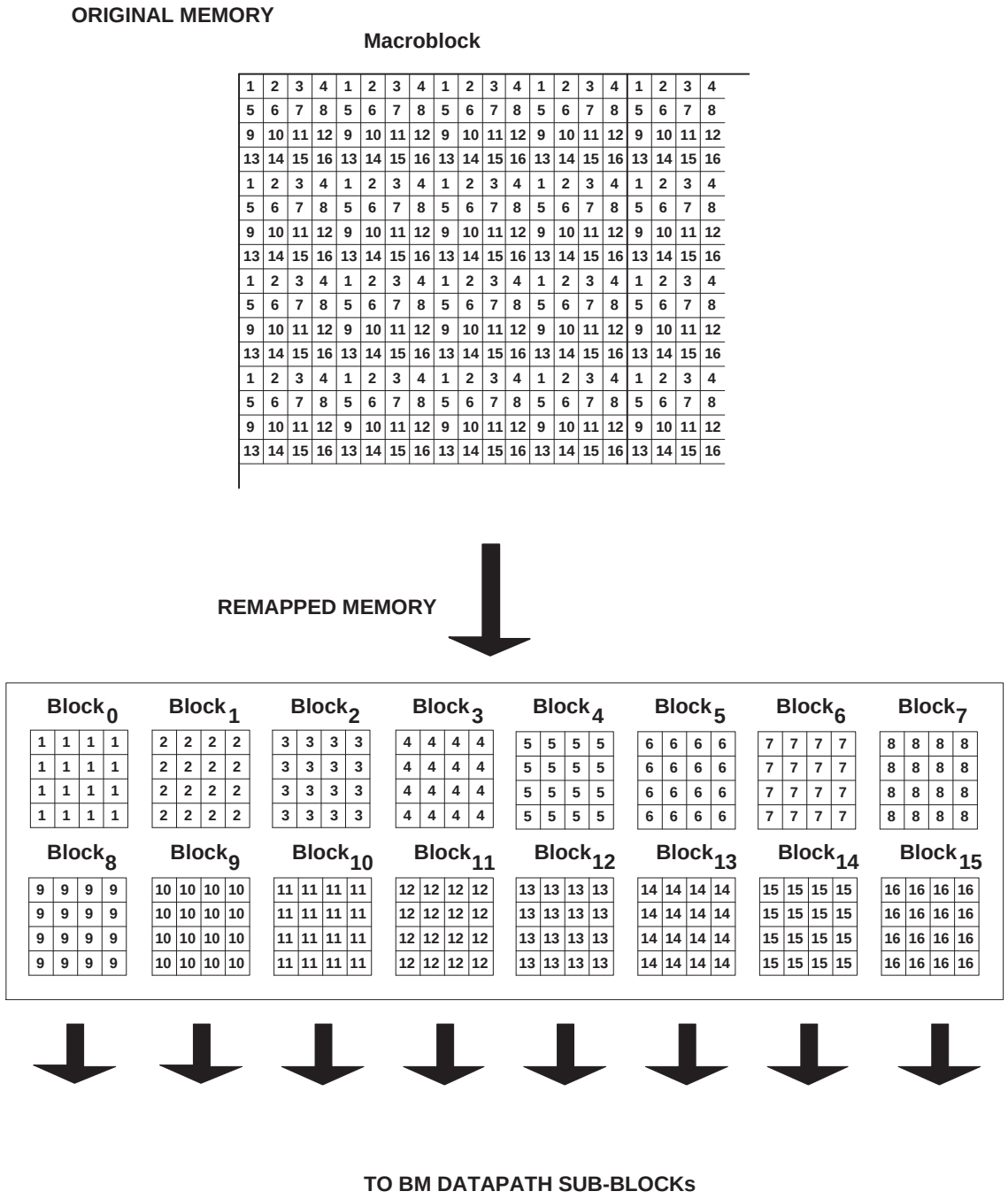


Figure 7.12: Memory Subsampling for 16xPE Architecture

Another alternative is not to use run length coding at all. In each clock cycle 4,8 or 16 binary pixels could be retrieved from memory. Equation 7.6 could still be applied, however the scope for early termination is reduced and a greater number of redundant operations will occur. The area is reduced since the RLC memory and the associated encoding & decoding logic is not required. Furthermore the throughput is increased since a single processing element retrieves more than a single pixel at a time. This approach potentially could be integrated with a systolic array type architecture for memory bandwidth improvements, though at a cost of reduced early termination and less scope to exploit redundant operations.

Finally, an alternative approach for updating TOT_{ref_blk} is to pre-process the frame and store the number of white pixels for each sub-block. In this way, the first three operations could be combined in a single clock cycle, which would reduce the processing time by 66% for poor candidate block matches. Although, a disadvantage of this approach is that the latency of the design is increased by one frame.

7.5 Summary of Contributions

Binary motion estimation is a fundamental tool for semantic video object shape processing. It is also widely acknowledged that motion estimation is the most computationally demanding block in a video codec. Whilst the computational cost of binary motion estimation is lessened due to the reduction of bits per pixel, prior research has shown that binary motion estimation consumes over 90% of the total resources required for MPEG-4 shape coding, which itself is the second most demanding block in a MPEG-4 object based encoder [45]. It is reasonable to conclude then that a computationally constrained platform will struggle to meet the necessary throughput and energy efficiency requirements for binary motion estimation for shape processing. This provides the motivation for the dedicated energy efficient hardware architecture that was presented in this chapter. A comprehensive overview of hardware motion estimation architectures was provided in Section 7.1, this has a particular focus on binary motion estimation. A detailed description of the proposed architecture was then given in Section 7.2. Algorithmic level power savings were suggested in the form of SAD reformulation and early SAD termination. The proposed architecture was compared to the related research in terms of silicon area, throughput and switching activity in Section 7.3. The proposed architecture was shown to be competitive with the prior related research. Finally a number of possible ways of improving and extending the architecture were discussed in Section

7.4.

CHAPTER 8

Conclusions & Future Work

This chapter summarises the research goals of this thesis and discusses the results achieved in the previous chapters. In particular, Section 8.1 reviews the motivations for this research, Section 8.2 provides a summary of the key contributions of this thesis and finally, Section 8.3 summarises the possible avenues of future research and discusses open research questions.

8.1 Motivation for Research – A Summary

There is an overwhelming human desire for untethered communications, instant entertainment (music, video, pictures, news, information) and improved personal and professional productivity. These human factors coupled with advances in semiconductor technology and signal processing have contributed heavily toward the tremendous growth in sales of mobile devices. In addition to their ubiquity, the frequent design and marketing of mobile devices as aspirational premium priced style icons along with device personalisation are clear indications of how deeply ingrained mobile devices are in today's society [10][11]. Furthermore, improving computational power is also enabling application convergence (e.g. telephone, SMS, email, video conferencing, Internet browsing, gaming, digital camera, digital audio/video player, mobile TV, etc) on these devices [7][8][9]. Multimedia applications feature strongly on these convergent devices. This is not surprising since users are frequently motivated by factors such as a desire for graphically enriched communications, filling "dead time" by consuming multimedia content or by creating multimedia

content (pictures & video) via spontaneously capturing moments of personal significance.

Whilst mobile devices have the potential to offer an engaging multimedia experience for the user, numerous technical challenges remain to be solved before next generation mobile multimedia can become a reality. In terms of image and video data, the principal challenges could be considered to include a need for more advanced processing to bridge the so-called semantic gap between pixels and the higher order structural meaning of collections of pixels. As was discussed in Chapter 1 and Chapter 2 by processing pixel data in terms of the semantic objects present in the image/video, improvements can be made to all applications in the "life-cycle" of pixel data. There is also a need to overcome throughput and battery life issues enforced by the constrained resources available on mobile devices. This is particularly relevant for all image/video processing algorithms, as these are generally considered one of the most computationally demanding tasks, due to elaborate resource intense algorithms coupled with a requirement for high throughput performance, frequently with a real time constraint. Finally, improvements to the efficiency and robustness of wireless delivery of multimedia content are also desirable, however this is outside the scope of this thesis.

Unfortunately, the greatest obstacle to widespread adoption of semantic object based processing is the fact that ideal semantic object segmentation is an inherently ill-posed problem. This can be made tractable by constraining the solution space and focusing on specific application scenarios. However, even when the solution space is constrained, semantic object segmentation algorithms are inherently computationally complex. On a mobile device this computational complexity increase has the highly undesirable effect of diminishing real time performance and increasing power consumption.

The combination of the outlined issues are unfortunate, particularly as semantic object based processing could be considered an enabling technology for a plethora of novel multimedia applications on mobile devices. This has provided the research motivation for this thesis to explore viable implementation options for specific enabling technologies for semantic object segmentation and processing on mobile devices. The segmentation solution proposed deliberately focused on mid-level semantic objects and in particular the segmentation of the human face as a fundamental semantic object of benefit to users of mobile devices. Subsequent processing of the segmented object was then considered in order to identify other bottlenecks which hinder the adoption of object based frameworks such as MPEG-4. From this analysis, it was concluded that motion estimation for shape is a fundamental task which requires considerable computational resources

and as such warranted further research. A summary of the research results achieved for the face segmentation/detection and motion estimation is outlined in the following section.

8.2 Summary of Thesis contributions

Chapter 1 gives a broad context for the research work described in this thesis. This context, which is summarised in Section 8.1, has the primary theme that although semantic video object based processing has many undefinable benefits, the principal issues that must be tackled before it can become viable on mobile devices are the object segmentation challenge and increased power consumption due to additional algorithmic processing of object based video. The goals and objectives of the thesis are also outlined in Chapter 1, followed by a description of the thesis structure. A more technical context for the research work is given in Chapter 2. A detailed overview of image and video compression theory is described along with the extensions necessary to support an object based paradigm. This section highlighted that MPEG-4 provides a complete framework for video object based compression and transport. It is also established that within MPEG-4, binary motion estimation for shape is the most computationally demanding task required to support object based compression. Implementation options for image/video processing algorithms are then discussed and it is concluded that dedicated hardware offers the highest throughput and best energy efficiency, but at a cost of reduced flexibility and increased development time. The sources of power consumption are then described along with common power consumption minimisation techniques. It is highlighted that the greatest scope for reducing power consumption occurs at the system/algorithmic level, where there is more degrees of freedom to impact the energy efficiency.

Chapter 3 provided a comprehensive review of face detection algorithms. This guided the selection of an appropriate face detection algorithm for this research and also gave insights into possible modifications to improve the characteristics of the chosen algorithm. From this review, it was also concluded that a software only implementation would struggle for real-time performance on a mobile device. This observation was factored into the algorithm selection decision, in that algorithms suited to dedicated hardware offload were deemed to offer a more attractive overall solution with higher throughput and lower power consumption. This lead to the selection of a novel algorithm employing an EANN. This approach could be considered to be a variant of the widely used high performance ANN face detection algorithms but with a GA controlling the learning phase [84]. The GA offers the potential to find novel non-trivial topologies in the overall topology

design space. Furthermore, as NEAT (the chosen EANN library) uses a minimal topology as the evolutionary search starting point there is scope to find a minimum sized topology for any given problem, including face detection. Such a topology is highly beneficial from an energy efficiency perspective as it reduce the numbers of operations and thus also the reduces the power consumption. This was the primary reason for selecting the NEAT library for the face detection training.

Comprehensive details of the face detection training algorithm were presented in Chapter 4. This included information on the preprocessing of the training data, an overview of the NEAT library in the context of the face detection training, a discussion on the design decisions made and finally the overall face detection training framework. An iterative approach to training was then taken, allowing constant exploration of GA parameters. Comprehensive details were given for two distinct versions of training. In the first approach, the starting topology used the default minimal topology, whilst a seeded starting topology with hidden neurons present was used in the second training approach. The non-seed seeded topology was found to perform best and gave a final topology size considerably smaller than the seeded approach. The size of the best trained topology was very favourable compared to conventional ANN topologies.

A thorough description of the proposed face detection algorithm during normal operation was given in Chapter 5. This included a discussion of the design decisions made in selecting the classifier ensemble and face detection merging. Profiling was then carried out, this verified the design goal that the hardware implementation amenable EANN evaluation and the associated tasks of loading the inputs and reseting the topology) are the most computationally demanding, requiring almost 95% of the total computational load of the algorithm. In addition to function-based profiling, run-time evaluation and power profiling results were presented for an ARM-920T microprocessor. This identified that a software only implementation of the proposed algorithm on a typical mobile processor would struggle for appropriate performance. This provides strong motivation for a dedicated hardware accelerator. Prior to this the software was optimised as much as possible by reducing the number of candidate face locations classified using the EANN. This was addressed by using minimum sized faces and skin detection pre-filtering of each candidate face location. Further software optimisation appropriate for a mobile device software implementation were also discussed. The trained face detection algorithm was shown to perform well on a dataset which could be considered representative of the image quality and facial pose found on the target mobile device application.

Following the conclusion in Chapter 5 that a software only implementation of the proposed face detection algorithm would struggle for acceptable performance, Chapter 6 presented the design of a dedicated hardware architecture for EANN acceleration. Rather than use a systolic array architecture, a SIMD datapath type structure was chosen. The motivation for this decision was that the efficiency of a systolic array is reduced by sparse EANN topologies, which is exactly the type of evolved topology generated by NEAT. In addition, the proposed architecture is flexible to handle any evolved architecture. The dedicated hardware architecture used a novel activation function generator. This was based upon a first order Minimax polynomial and was further optimised using a genetic algorithm. The activation function generator was implemented using efficient fused multiply add circuitry. The overall architecture was shown to compare favourably to the prior art in terms of throughput, silicon resources and power consumption. Furthermore, the proposed architecture was shown to be greatly superior to a software implementation on an ARM 920T processor (commonly found on mobile devices), in terms of run time and energy efficiency.

Assuming a video object such as a face is segmented, possible bottlenecks in subsequent processing were then considered. Motion estimation was then identified as being a primary bottleneck in the adoption of a video object based framework such as MPEG-4. Consequently, a novel architecture for binary motion estimation was proposed. The proposed binary motion estimation architecture reduces power consumption through early termination of SAD calculation and by minimising access to redundant binary data. This was facilitated by a reformulation of the block matching SAD distortion metric. The architecture was shown to be competitive with the prior art, trading off throughput for reduced power consumption.

8.3 A Vision for the Future

Short term techniques for improving the research presented in this thesis were generally discussed at the conclusion of each relevant chapter. Taking a longer term viewpoint, there are two principal areas that warrant prolonged future research. The first of these is a need for greater robustness in the face detection performance, particularly invariance to pose and viewpoint of the face. The research to date in the literature in this area has focused on using multiple classifiers trained for different poses/viewpoints. The challenge for a computationally constrained device is how this can be done in an energy efficient manner. Secondly, the memory architectures for both the EANN and binary motion estimation hardware architectures are worth investigating further in order to increase

throughput. As with many image and video processing algorithms, the challenge is frequently how to overcome the data/memory bound problem and do so in a energy efficient manner. This is also an issue faced by designers of emerging highly parallel multiple core and array processors.

Looking outside of the author's core research, it could be said that EANN algorithms have excellent potential for discovering novel solutions to many difficult real world problems. However, before they can be used more widely for image and video processing, the issue of searching and finding suitable topologies in very large dimensional search spaces (that are inherent with image/video data) need to be further addressed. Another challenge for the research community is accurate and fair benchmarking. A fundamental issue encountered in this thesis was a lack of power consumption and energy results to compare against. This challenge extends beyond merely recording additional results, there is a requirement for technology independent metrics. For example, in the absence of reimplementing face detection algorithms (which would be excessively time consuming) benchmarking performance on mobile devices is challenging and this situation will only become worse in the future when multiple core processors could further hide the complexity of one algorithm over another. Comparing run time throughput performance is not particularly useful since no indication of the processor clock frequency or architecture is given. Quoting clock frequency is marginally better, but does require scaling which is imprecise at best. Ideally what is required are metrics that are independent of design tool performance, processor architecture, clock frequency, silicon process and any operating system loading.

Bibliography

[1] G. E. Moore, “Cramming More Components onto Integrated Circuits,” *Electronics*, vol. 38, no. 8, Apr. 19 1965.

[2] Intel Corporation. Moore’s Law. [Online]. Available: http://www.intel.com/technology/mooreslaw/index.htm

[3] Gartner. Gartner Says Top Six Vendors Drive Worldwide Mobile Phone Sales to 21% Growth in 2005. [Online]. Available: http://www.gartner.com/press_releases/asset_145891_11.html

[4] J. Walko. (2006, Jan.2,) Record year for mobile phone shipments – analysts. [Online]. Available: http://www.eet.com/news/latest/showArticle.jhtml?articleID=177104183

[5] M. Kanellos. (2005, Oct.) Pc shipments leap past expectations. [Online]. Available: http://news.com.com/PC+shipments+leap+past+expectations/2100-1003_3-5897839.html?tag=st.bp.story

[6] J. Walko. (2007, Jan.) After 23 years, mobile phone sales breach billion mark. [Online]. Available: http://www.eetimes.eu/uk/197000427

[7] T. Austin, D. Blaauw, S. Mahlke, T. Mudge, C. Chakrabarti, and W. Wolf, “Mobile supercomputers,” *IEEE Computer*, vol. 37, pp. 81–83, May 2004.

[8] P. Mannion. (2003, 26th March) Advanced features set to drive handset replacements, chipsets. [Online]. Available: http://www.commsdesign.com/showArticle.jhtml?articleID=16500574

[9] R. Wilson. (2004, July) That sucking sound? it's your cell. [Online]. Available: http://www.eetimes.com/news/latest/showArticle.jhtml?articleID=25600132&printable=true

[10] EE Times. (2006, Jul. 10,) Nokia's 7280 puts fashion first. [Online]. Available: http://www.eetimes.com/news/design/showArticle.jhtml?articleID=190300355

[11] Apple. (2006, Sep.12,) Apple unveils the new ipod. [Online]. Available: http://www.apple.com/pr/library/2006/sep/12ipod.html

[12] Reuters. (2006, May 17,) Ringtone spin-offs boost mobile music. [Online]. Available: http://news.cnet.co.uk/mobiles/0,39029678,49272725,00.htm

[13] S. Shim. (2006, Aug.22,) Samsung offers mobile phone with 8-gig hard disk. [Online]. Available: www.eetimes.com/news/latest/showArticle.jhtml?articleID=192202809

[14] BBC. (2003, Feb.) Video recorders enter the digital age. [Online]. Available: http://news.bbc.co.uk/1/hi/technology/2762579.stm

[15] J. Twist. (2006, Jan.) The year of the digital citizen. [Online]. Available: http://news.bbc.co.uk/1/hi/technology/4566712.stm

[16] D. Molta. (2006, Sep.29,) Mobile Video: The Next Killer App. [Online]. Available: http://www.mobilehandsetdesignline.com/news/193101121

[17] Flickr online photo sharing. [Online]. Available: http://www.flickr.com

[18] You Tube - online video sharing. [Online]. Available: http://www.youtube.com

[19] J. Arreymbi and M. Dastbaz, "Issues in delivering multimedia content to mobile devices," in *Information Visualisation, 2002. Proceedings. Sixth International Conference on*, Jul. 2002, pp. 622–626.

[20] N. Banerjee, A. Acharya, and S. Das, "Seamless SIP-based mobility for multimedia applications," *IEEE Network*, vol. 20, no. 2, pp. 6–13, Mar./Apr. 2006.

[21] C. Cicconetti, L. Lenzini, E. Mingozzi, and C. Eklund, "Qality of service support in IEEE 802.16 networks," *IEEE Network*, vol. 20, no. 2, pp. 50–55, Mar./Apr. 2006.

[22] J. Li and H.-H. Chen, "Mobility support for IP-based networks," *IEEE Communications Magazine*, vol. 43, no. 10, pp. 127–132, Oct. 2005.

[23] X. Fu, H. Schulzrinne, A. Bader, D. Hogrefe, C. Kappler, G. Karagiannis, H. Tschofenig, and S. Van den Bosch, "NSIS: a new extensible IP signaling protocol suite," *IEEE Communications Magazine*, vol. 43, no. 10, pp. 133–141, Oct. 2005.

[24] S. Rai, B. Mukherjee, and O. Deshpande, "IP resilience within an autonomous system: current approaches, challenges, and future directions," *IEEE Communications Magazine*, vol. 43, no. 10, pp. 142–149, Oct. 2005.

[25] (2000, August) A wap on the knuckles. [Online]. Available: http://www.uidesign.net/2000/opinion/wapknuckles.html

[26] (2001, June) Wap usage nosedives in the nordic countries. [Online]. Available: http://wbt.sys-con.com/read/40845.htm

[27] A. Albiol, C. A. Bouman, and E. J. Delp, "Face detection for pseudo-semantic labeling in video databases," in *Image Processing, 1999. ICIP 99. Proceedings. 1999 International Conference on*, vol. 3, Kobe, Oct. 1999, pp. 607–611.

[28] M. Roach, J. Mason, L. Xu, and F. Stentiford, "Recent trends in video analysis : a taxonomy of video classification problems," in *In Proceedings of the International Conference on Internet and Multimedia Systems and Applications*, Aug. 2002.

[29] A. G. Hauptmann, "Lessons for the Future from a Decode of Informedia Video Analysis Research," in *Proceedings of the 4th International Conference on Image and Video Retrieval*, Jul. 2005, pp. 1–10.

[30] B. Widrow, D. E. Rumelhart, and M. A. Lehr, "Neural Networks: Applications in Industry, Business and Science," *Communications of the ACM*, vol. 37, no. 3, pp. 93–105, Mar. 1994.

[31] K. Stanley, B. Bryant, and R. Miikkulainen, "Real-time neuroevolution in the NERO video game," *IEEE Transactions on Evolutionary Computation*, vol. 9, no. 6, pp. 653–668, Dec. 2005.

[32] P. M. Kuhn, *Algorithms, Complexity Analysis and VLSI Architectures for MPEG-4 Motion Estimation*. Springer, Jun. 1999.

[33] C. A. Rahman and W. Badawy, "A quarter pel full search block motion estimation architecture for h.264/AVC," in *Multimedia and Expo, 2005. ICME 2005. IEEE International Conference on*, Jul. 2005.

[34] P. Symes, *Video Compression Demystified*. McGraw-Hill Education, February 2001.

[35] I. E. Richardson, *H.264 and MPEG-4 Video Compression*. Wiley, 2004.

[36] T. Xanthopoulos and A. P. Chandrakasan, "A Low-Power IDCT Macrocell for MPEG-2 MP@ML Exploiting Data Distribution Properties for Minimal Activity," *IEEE Journal of Solid-State Circuits*, vol. 34, no. 9, pp. 693–703, May 1999.

[37] A. Kinane, "Energy Efficient Hardware Acceleration of Multimedia Processing Tools," Ph.D. dissertation, School of Electronic Engineering, Dublin City University, Apr. 2006. [Online]. Available: http://www.eeng.dcu.ie/~kinanea/thesis/kinane_final.pdf

[38] M. Ghanbari, *Video Coding, an Introduction to standard codecs*. IEE, 1999.

[39] K. Jack, *Video Demystified*. HighText Publications, 2005.

[40] *Coding of audio-visual objects – Part 10: Advanced Video Coding*, ISO/IEC 14496-10:2005 Std., Rev. 3rd Edition, 2007. [Online]. Available: http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=43058

[41] G. Sullivan and T. Wiegand, "Video compression - from concepts to the h.264/avc standard," *Proceedings of the IEEE*, vol. 93, no. 1, pp. 18 – 31, Jan. 2005.

[42] *MPEG-4: Information Technology – Coding of Audio Visual Objects – Part 2: Visual, ISO/IEC 14496-2*, ISO/IEC Std., Rev. Amendment 1, Jul. 2000.

[43] T. Sikora and B. Makai, "Shape-Adaptive DCT for Generic Coding of Video," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 5, no. 1, pp. 59–62, Feb. 1995.

[44] N. Brady, "MPEG-4 Standardized Methods for the Compression of Artibitarily Shaped Video Objects," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 9, no. 8, Dec. 1999.

[45] P.-C. Tseng, Y.-C. Chang, Y.-W. Huang, H.-C. Fang, C.-T. Huang, and L.-G. Chen, "Advances in Hardware Architectures for Image and Video Coding – A Survey," *Proceedings of the IEEE*, vol. 93, no. 1, pp. 184–197, Jan. 2005.

[46] N. O'Connor and N. Murphy. (2003) Image and Video Compression – EE554 Taught M.Eng. Course Notes. Dublin City University. [Online]. Available: http://www.eeng.dcu.ie/~oconnorn

[47] O. J. Morris, M. J. Lee, and A. G. Constantinides, "Graph theory for image analysis: an approach based on the shortest spanning tree," *IEE Proceedings*, vol. 133, pp. 146–152, Apr. 1986.

[48] T. Meier and K. N. Ngan, "Automatic Segmentation of Moving Objects for Video Object Plane Segmentation," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 8, no. 5, Sep. 1998.

[49] J. Fan, D. K. Yau, A. K. Elmagarmid, and W. G. Aref, "Automatic Image Segmentation by Integrating Color-Edge Extraction and Seeded Region Growing," *IEEE Transactions on Image Processing*, vol. 10, no. 10, Oct. 2001.

[50] C. Gu and M.-C. Lee, "Semiautomatic Segmentation and Tracking of Semantic Video Objects," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 8, no. 5, Sep. 1998.

[51] T. Meier and K. N. Ngan, "video Segmentation for Content-Based Coding," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 9, no. 8, Dec. 1999.

[52] C. Kim and J.-N. Hwang, "Fast and Automatic Video Object Segmentation and Tracking for Content-Based Applications," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 12, no. 2, Feb. 2001.

[53] S. Sun, D. R. Haynor, and Y. Kim, "Fast and Automatic Video Object Segmentation and Tracking for Content-Based Applications," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 13, no. 1, Jan. 2003.

[54] M. Meribout and M. Nakanishi, "A New Real Time Object Segmentation and Tracking Algorithm and its Parallel Harware Architecture," *Springer Journal of VLSI for Signal Processing*, vol. 39, no. 3, Mar. 2005.

[55] J. Stauder, G. Gouzien, B. Chupeau, J. Vigouroux, and E. Kijak, "Semantic image browsing using hidden categories and confidence values," in *Electronic Imaging 2003*, Jan. 2003.

[56] D. A. Sadlier and N. E. O'Connor, "Event detection in field sports video using audio-visual features and a support vector machine," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 15, no. 10, pp. 1225–1233, Oct. 2005.

[57] B. Lehane, N. O'Connor, and H. Lee., "Searching movies based on user defined semantic events," in *SIGMAP 2006 - International Conference on Signal Processing and Multimedia Applications*, Aug. 7–10, 2006, pp. 232–239.

[58] P. Pirsch and H.-J. Stolberg, "VLSI Implementations of Image and Video Multimedia Processing Systems," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 8, no. 7, pp. 878–891, Nov. 1998.

[59] A. Dasu and S. Panchanathan, "A Survey of Media Processing Approaches," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 12, no. 8, pp. 633–645, Aug. 2002.

[60] B. Hori and J. Eyre. (2005, Mar. 14,) Inside DSP on Digital Video: Processors for Video. [Online]. Available: http://insidedsp.eetimes.com/features/showArticle.jhtml?articleID=159401710

[61] Intel Corporation. [Online]. Available: http://www.intel.com/

[62] S. W. Smith, *Digital Signal Processing – A Practical Guide for Engineers and Scientists*, 1st ed. Newnes, 2003.

[63] Xtensa Product Overview. Tensilica. [Online]. Available: http://www.tensilica.com/html/products.html

[64] B. Cope, P. Y. K. Cheung, W. Luk, and S. Witt, "Have GPUs made FPGAs redundant in the field of video processing?" in *Field-Programmable Technology, 2005. Proceedings. 2005 IEEE International Conference on*, Dec. 11–14, 2005, pp. 111–118.

[65] A. P. Chandrakasan and R. W. Brodersen, *Low Power Digital CMOS Design*, 1st ed. Dordrecht The Netherlands: Kluwer Academic Publishers, 1995.

[66] K. Roy and S. C. Prasad, *Low-Power CMOS VLSI Circuit Design*. John Wiley and Sons, 2000.

[67] J. M. Rabaey and M. Pedram, *Low Power Design Methologies*. Kluwer Academic Publishers, 1996.

[68] M. Pedram and J. M. Rabaey, *Power Aware Design Methologies*. Kluwer Academic Publishers, 2002.

[69] W. Elgharbawy and M. Bayoumi, "Leakage Sources and Possible Solutions in Nanometer CMOS Technologies," *IEEE Circuits and Systems Magazine*, vol. 5, pp. 6–17, Dec. 2005.

[70] K. Morris. (2005, April 26) Power - Suddenly, We care. [Online]. Available: http://www.fpgajournal.com/articles_2005/20050426_power.htm

[71] A. Chandrakasan and R. Brodersen, "Minimizing power consumption in digital cmos circuits," *Proceedings of the IEEE*, vol. 83, pp. 498–523, Apr. 1995.

[72] (2004, July 15) Minimize IC power without sacrificing performance. URL: http://www.eedesign.com/. [Online]. Available: http://www.eedesign.com/article/showArticle.jhtml?articleId=23901143

[73] The Intel Pentium M Processor: Microarchitecture and Performance: ENHANCED INTEL SPEEDSTEP TECHNOLOGY. [Online]. Available: http://www.intel.com/technology/itj/2003/volume07issue02/art03_pentiumm/p10_speedstep.htm

[74] ARM Homepage. [Online]. Available: http://www.arm.com/

[75] Advanced Micro Devices, AMD - Homepage. [Online]. Available: http://www.amd.com/

[76] I. Koren, *Computer Arithmetic Algorithms*. A K Peters Ltd, 2001.

[77] Digital Photography Review. (2006, Sep. 14,) Canon's first wide angle Digital IXUS includes DIGIC III and Face Detection. [Online]. Available: http://www.dpreview.com/news/0609/06091403_canon_sd800is.asp

[78] Fujifilm. (2006, Sep. 25,) Fujifilm brings face detection to the compact digital camera category with the finepix f31fd. [Online]. Available: http://www.cameratown.com/news/news.cfm/hurl/id%7C3106

[79] fotonation. Fotonation face tracker. [Online]. Available: http://www.fotonation.com/index.php?module=products&id=55

[80] M.-H. Yang, D. 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, Jan. 2002.

[81] E. Hjelmas and B. K. Low, "Face Detection: A Survey," *Computer Vision and Understanding*, vol. 83, no. 3, pp. 236–274, Sep. 2001.

[82] M. J. Er, W. Chen, and S. Wu, "High-speed face recognition based on discrete cosine transform and rbf neural network," *IEEE Transactions on Neural Networks*, vol. 16, no. 3, pp. 679–691, May 2005.

[83] C. Garcia and M. Delakis, "Convolutional face finder: a neural architecture for fast and robust face detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 26, no. 11, pp. 1408 – 1423, Nov. 2004.

[84] H. Rowley, S. Baluja, and T. Kanade, "Neural network-based face detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 20, no. 1, pp. 23 – 38, Jan. 1998.

[85] R.-L. Hsu, M. Abdel-Mottaleb, and A. K. Jain, "Face detection in color images," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 24, no. 5, pp. 696–706, May 2002.

[86] O. Ikeda, "Segmentation of faces in video footage using HSV color for face detection and image retrieval," *2003. ICIP 2003. Proceedings. 2003 International Conference on Image Processing*, vol. 3, pp. 913–6, Sep. 2003.

[87] J. C. Terrillon, M. N. Shirazi, H. Fukamachi, and S. Akamatsu, "Comparative performance of different skin chrominance models andchrominance spaces for the automatic detection of human faces in colorimages," in *Automatic Face and Gesture Recognition, 2000. Proceedings. Fourth IEEE International Conference on*, Grenoble, France, 2000, pp. 54–61.

[88] M. J. Jones and J. M. Rehg, "Statistical color models with application to skin detection," in *Computer Vision and Pattern Recognition, 1999. IEEE Computer Society Conference on.*, vol. 1, Fort Collins, CO, USA, 1999.

[89] J. Kovac, P. Peer, and F. Solina, "Human skin color clustering for face detection," in *EUROCON 2003. Computer as a Tool. The IEEE Region 8*, vol. 2, Sep. 2003, pp. 144–148.

[90] S. Cooray and N. O'Connor, "Facial features and appearance-based classification for face detection in color images," in *Proceedings of the 11th International Workshop on Systems, Signals and Image Processing, IWSSIP'04*, Poznan, Poland, Sep. 2004.

[91] M. Hunke and A. Waibel, "Face locating and tracking for human-computer interaction," in *Signals, Systems and Computers, 1994. 1994 Conference Record of the Twenty-Eighth Asilomar Conference on*, vol. 2, Pacific Grove, CA, USA, Oct./Nov. 1994, pp. 1277–1281.

[92] J. Yang and A. Waibel, "A real-time face tracker," in *Applications of Computer Vision, 1996. WACV '96., Proceedings 3rd IEEE Workshop on*, Sarasota, FL, Dec. 1996, pp. 142–147.

[93] D.-J. Niu, Y.-Z. Zhan, and S.-L. Song, "Research and implementation of real-time face detection, tracking and protection," in *Machine Learning and Cybernetics, 2003 International Conference on*, vol. 5, Nov. 2003, pp. 2765–2770.

[94] Y.-X. Lv, Z.-Q. Liu, and X.-H. Zhu, "Real-time face detection based on skin-color model and morphology filters," in *Machine Learning and Cybernetics, 2003 International Conference on*, vol. 5, Nov. 2003, pp. 3203–3207.

[95] N. A. Khan and A. Z. Muhammad Hassan Khan, "Reliable face detection in natural scenes," in *Proceedings of the 11th International Workshop on Systems, Signals and Image Processing, IWSSIP'04*, Poznan, Poland, Sep. 2004.

[96] D. Chai and K. N. Ngan, "Face segmentation using skin-color map in videophone applications," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 9, no. 4, pp. 551–564, Jun. 1999.

[97] M.-J. Chen, M.-C. Chi, C.-T. Hsu, and J.-W. Chen, "ROI video coding based on h.263+ with robust skin-color detection technique," in *Consumer Electronics, 2003. ICCE. 2003 IEEE International Conference on*, Jun. 2003, pp. 44–45.

[98] L. Zhi-fang, Y. Zhi-sheng, A. K. Jain, and W. Yun-qiong, "Face detection and facial feature extraction in color image," in *Computational Intelligence and Multimedia Applications, 2003. ICCIMA 2003. Proceedings. Fifth International Conference on*, Sep. 2003, pp. 126–130.

[99] S. L. Phung, A. Bouzerdoum, and D. Chai, "A novel skin color model in YCbcr color space and its application to human face detection," *Image Processing. 2002. Proceedings. 2002 International Conference on*, vol. 1, 2002.

[100] C.-W. Lin, Y.-J. Chang, and Y.-C. Chen, "Low-complexity face-assisted video coding," in *Image Processing, 2000. Proceedings. 2000 International Conference on*, vol. 2, Vancouver, BC, Canada, 2000, pp. 207–210.

[101] N. Tsapatsoulis, Y. Avrithis, and S. Kollias, "Efficient face detection for multimedia applications," in *Proceedings of the IEEE International Conference on Image Processing*, vol. 2, Vancouver, BC, Canada, Sep. 2000, pp. 247–250.

[102] S. Cooray and N. O'Connor., "Facial feature extraction and principal component analysis for face detection in color images," in *ICIAR 2004 - International Conference on Image Analysis and Recognition*, 2004.

[103] C.-C. Han, H.-Y. M. Liao, G.-J. Yu, and L.-H. Chen, "Fast face detection via morphology-based pre-processing," in *ICIAP '97: Proceedings of the 9th International Conference on Image Analysis and Processing-Volume II*. London, UK: Springer-Verlag, 1997, pp. 469–476.

[104] W. Huang, Q. Sun, C.-P. Lam, and J.-K. Wu, "A robust approach to face and eyes detection from images with cluttered background," in *Pattern Recognition, 1998. Proceedings. Fourteenth International Conference on*, vol. 1, Brisbane, Qld., Aug. 1998, pp. 110–113.

[105] B. Takacs and H. Wechsler, "Face location using a dynamic model of retinal feature extraction," in *International Workshop on Automatic Face and gesture Recognition*, Zurich, Switzerland, 1995, pp. 243–247.

[106] R. Herpers, M. Michaelis, K. H. Lichtenauer, and G. Sommer, "Edge and keypoint detection in facial regions," in *Automatic Face and Gesture Recognition, 1996., Proceedings of the Second International Conference on*, Killington, VT, Oct. 1996, pp. 212–217.

[107] F. Smeraldi, O. Carmona, and J. Bigun, "Saccadic search with gabor features applied to eye detection and real-time head tracking," *Elsevier Image and Vision Computing*, no. 18, pp. 323–329, 2000.

[108] T. Yokoyama, Y. Yagi, and M. Yachida, "Active contour model for extracting human faces," in *Pattern Recognition, 1998. Proceedings. Fourteenth International Conference on*, vol. 1, Brisbane, Qld., Aug. 1998, pp. 673–676.

[109] S. Paschalakis and M. Bober, "A low cost fpga system for high speed face detection and tracking," in *Proceedings of the IEEE International Conference on Field-Programmable Technology (FPT)*, Tokyo, Japan, Dec. 2003, pp. 214–221.

[110] H. Wu, Q. Chen, and M. Yachida, "Face detection from color images using a fuzzy pattern matching method," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 21, no. 6, pp. 557–563, Jun. 1999.

[111] 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, Sep. 2004.

[112] C. Huang, H. Ai, Y. Li, and S. Lao, "High-performance rotation invariant multiview face detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 29, no. 4, pp. 671–686, Apr. 2007.

[113] P. Viola and M. Jones, "Robust real-time object detection," in *Proceedings of the Second International Workshop on Statistical and Computational theories of Vision – Modeling, Learning, Compututing and Sampling*, Jul.13, 2001.

[114] M. Turk and A. Pentland, "Eigenfaces for recognition," *Journal of Cognitive Neuroscience*, vol. 3, no. 1, pp. 72–86, 1991.

[115] K.-K. Sung and T. Poggio, "Example-Based Learning for View-Based Human Face Detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 20, pp. 39 – 51, Jan. 1998.

[116] R. Feraud, O. J. Bernier, J.-E. Viallet, and M. Collobert, "A fast and accurate face detector based on neural networks," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 23, no. 1, pp. 42–53, Jan. 2001.

[117] A. V. Nefian and I. Hayes, M. H., "Face detection and recognition using hidden markov models," in *Image Processing, 1998. ICIP 98. Proceedings. 1998 International Conference on*, vol. 1, Chicago, IL, USA, Oct. 1998, pp. 141–145.

[118] A. V. Nefian and M. H. H. III, "Maximum likelihood training of the embedded hmm for face detection and recognition," in *Proceedings of the IEEE International Conference on Image Processing*, vol. 1, Sep. 2000, pp. 33 – 36.

[119] P. Viola and M. Jones, "Fast multi-view face detection," Mitsubishi Electric Research Laboratories, Tech. Rep., Jul. 2003.

[120] H. Schneiderman and T. Kanade., "Probabilistic Modelling of Local Appearance and Spatial Relationships for Object Recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, July 1998, pp. 45–51.

[121] H. Schneiderman and T. Kanade, "A statistical method for 3d object detection applied to faces and cars," in *Computer Vision and Pattern Recognition, 2000. Proceedings. IEEE Conference on*, vol. 1, Hilton Head Island, SC, Jun. 2000, pp. 746–751.

[122] C. Liu, "A bayesian discriminating features method for face detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 25, no. 6, pp. 725–740, Jun. 2003.

[123] E. Osuna, R. Freund, and F. Girosi., "Training support vector machines: An application to face detection," in *IEEE Proc. of Int. Conf. on Computer Vision and Pattern Recognition*, San Juan, Puerto Rico, Jun. 1997, pp. 130–136.

[124] S. Romdhani, P. Torr, B. Scholkopf, and A. Blake, "Computationally efficient face detection," in *Computer Vision, 2001. ICCV 2001. Proceedings. Eighth IEEE International Conference on*, vol. 2, Vancouver, BC, Canada, 2001, pp. 695–700.

[125] C. A. Waring and X. Liu, "Face detection using spectral histograms and SVMs," *Systems, Man and Cybernetics, Part B, IEEE Transactions on*, vol. 35, no. 3, pp. 467–476, Jun. 2005.

[126] M. Kirby and L. Sirovich, "Application of the karhunen-loeve procedure for thecharacterization of human faces," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 12, no. 1, pp. 103–108, Jan. 1990.

[127] C. S. S. Prasanna, N. Sudha, and V. Kamakoti, "A principal component neural network-based face recognition system and ASIC implementation," *2005. 18th International Conference on VLSI Design*, pp. 795–798, Jan. 2005.

[128] A. N. Rajagopalan, K. S. Kumar, J. Karlekar, R. Manivasakan, M. M. Patil, U. B. Desai, P. G. Poonacha, and S. Chaudhuri, "Finding faces in photographs," in *Computer Vision, 1998. Sixth International Conference on*, Bombay, India, Jan. 1998, pp. 640–645.

[129] P. Phillips, H. Moon, S. Rizvi, and P. Rauss, "The feret evaluation methodology for face-recognition algorithms," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, no. 10, pp. 1090–1104, Oct. 2000.

[130] H. Rowley, S. Baluja, and T. Kanade, "Rotation invariant neural network-based face detection," in *Proceedings of the 1998 IEEE International Conference on Computer Vision and Pattern Recognition*, Jun. 1998, pp. 38–44.

[131] D. B. Fogel, L. J. Fogel, and V. W. Porto, "Evolving neural networks," *Biol. Cybern.*, vol. 63, no. 6, pp. 487–493, 1990.

[132] X. Yao, "Evolving artificial neural networks," *Proceedings of the IEEE*, vol. 87, no. 9, pp. 1423– 1447, Sep. 1999.

[133] K. O. Stanley and R. Miikkulainen, "Evolving neural networks through augmenting topologies," *Evol. Comput.*, vol. 10, no. 2, pp. 99–127, 2002.

[134] S. Wiegand and C. Igel, "Evolutionary multi-objective optimization of neural networks for face detection," *Journal of Computational Intelligence and Applications*, vol. 4, no. 3, pp. 1–15, 2004.

[135] J. Montgomery, "Detecting faces using evolution," Master's thesis, Dept. of Computer Science, Birmingham University, UK, 2004.

[136] M. Yang, D. Roth, and N. Ahuja, "A SNoW-based face detector," in *Adv. Neural Information Process. Syst.*, vol. 12, 2000.

[137] F. Samaria and S. Young, "Hmm-based architecture for face identification," *Image and vision computing*, vol. 12, pp. 537–543, Oct. 1994.

[138] J. Šochman and J. Malas, "Adaboost with totally corrective updates for fast face detection," in *Automatic Face and Gesture Recognition, 2004. Proceedings. Sixth IEEE International Conference on*, May 2004, pp. 445–450.

[139] B. Wu, H. Ai, C. Huang, and S. Lao, "Fast rotation invariant multi-view face detection based on real adaboost," in *Automatic Face and Gesture Recognition, 2004. Proceedings. Sixth IEEE International Conference on*, May 2004, pp. 79–84.

[140] R. Etienne-Cummings, P. Pouliquen, and M. A. Lewis, "A vision chip for color segmentation and pattern matching," *EURASIP Journal on Applied Signal Processing*, pp. 700–712, Jan. 2003.

[141] F. Boussaid, D. Chai, and A. Bouzerdoum, "On-chip skin detection for color CMOS imagers," in *Proceedings of the International Conference on MEMS, NANO and Smart Systems, 2003*, Banff, Alberta - Canada, July 2003, pp. 357 – 361.

[142] M. Krips, T. Lammert, and A. Kummert, "FPGA implementation of a neural network for a real-time hand tracking system," in *Electronic Design, Test and Applications, 2002. Proceedings. The First IEEE International Workshop on*, Christchurch, Jan. 2002, pp. 313–317.

[143] T. Theocharides, G. Link, N. Vijaykrishnan, M. Irwin, and W. Wolf, "Embedded hardware face detection," in *Proceedings. 17th International Conference on VLSI Design, 2004*, Mumbai, India, Jan. 2004, pp. 133 – 138.

[144] M. S. Sadri, N. Shams, M. Rahmaty, I. Hosseini, R. Changiz, S. Mortazavian, S.Kheradmand, and R. Jafari, "An fpga based fast face detector," in *Global Signal Processing Expo and Conference (GSPX'04)*, Santa Clara, CA, Jul. 2004.

[145] F. Smach, M. Atri, J. Mitéran, and M. Abid, "Design of a neural networks classifier for face detection," *Journal of Computer Science*, vol. 2, no. 3, pp. 257–260, 2006.

[146] D. Nguyen, D. Halupka, P. Aarabi, and A. Sheikholeslami, "Real-time face detection and lip feature extraction using field-programmable gate arrays," *Systems, Man and Cybernetics, Part B, IEEE Transactions on*, vol. 36, no. 4, pp. 902–912, Aug. 2006.

[147] Y. Suzuki and T. Shibata, "Illumination-invariant face identification using edge-based feature vectors in pseudo-2d hidden markov models," in *European Signal Processing Conference*, Florence, Italy, Sep.14, 2006.

[148] M. Yang, Y. Wu, J. Crenshaw, B. Augustine, and R. Mareachen, "Face detection for automatic exposure control in handheld camera," in *Computer Vision Systems, 2006 ICVS '06. IEEE International Conference on*, Jan. 2006, pp. 17–17.

[149] T. Nakano, T. Morie, and A. Iwata, "A face/object recognition system using FPGA implementation of coarse region segmentation," in *SICE 2003 Annual Conference*, vol. 2, Aug. 2003, pp. 1552–1557.

[150] R. Chellappa, S. S. Bhattacharyya, S. Saha, W. Wolf, G. Aggarwal, J. Schlessman, and V. Kianzad, "An architectural level design methodology for embedded face detection," in *Hardware/Software Codesign and System Synthesis, 2005. CODES+ISSS '05. Third*

IEEE/ACM/IFIP International Conference on, Jersey City, NJ, USA, Sep. 2005, pp. 136–141.

[151] R. Genov and G. Cauwenberghs, "Kerneltron: Support vector machine in silicon," *IEEE Transactions on Neural Networks*, vol. 14, no. 5, pp. 1426–1434, Sep. 2003.

[152] F. Khan, M. Arnold, and W. Pottenger, "Hardware-based support vector machine classification in logarithmic number systems," in *IEEE International Symposium on Circuits and Systems 2005*, May 2005, pp. 5124 – 5127.

[153] M. Egmont-Petersen, D. De Ridder, and H. Handels, "Image processing with neural networks – a review," *Pattern Recognition*, vol. 35, pp. 2279–2301, 2002.

[154] I. R. Fasel and J. R. Movellan, "A comparison of face detection algorithms," in *International Conference on Artifical Neural Networks*, Madrid, Spain, Aug. 2002, pp. 1325–1330.

[155] L. Reyneri, "Implementation issues of neuro-fuzzy hardware: going toward hw/sw codesign," *IEEE Transactions on Neural Networks*, vol. 14, no. 1, pp. 176–194, Jan. 2003.

[156] K. O. Stanley, "Efficient evolution of neural networks through complexification," Ph.D. dissertation, The University of Texas, Austin, USA, Aug. 2004.

[157] L. Fausett, *Fundamentals of Neural Networks*. Prentice-Hall, 1994.

[158] S. Haykins, *Neural Networks - A Comprehensive Foundation*. Prentice-Hall, 1999.

[159] Z. Michalewicz, *Genetic Algorithms + Data Structures = Evolution Programs*, 3rd ed. New York: Springer, 1998.

[160] X. Yao and Y. Liu, "A new evolutionary system for evolving artificial neural networks," *IEEE Transactions on Neural Networks*, vol. 8, no. 3, pp. 694–713, May 1997.

[161] X. Yao, "Evolving artificial neural networks," *Proceedings of the IEEE*, vol. 87, no. 9, pp. 1423–1447, Sep. 1999.

[162] G. F. Miller, P. M. Todd, and S. U. Hegde, "Designing neural networks using genetic algorithms," in *Proceedings of the third international conference on Genetic algorithms*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1989, pp. 379–384.

[163] X. Yao, "Evolutionary artificial neural networks," *International Journal of Neural Systems (IJNS)*, vol. 4, no. 3, pp. 203–222, Oct. 1993.

[164] M. Buckland, *AI Techniques for Game Programming*, 1st ed. Premier Press, 2002.

[165] R. W. Frischholz and U. Dieckmann, "Biold: a multimodal biometric identification system," *Computer*, vol. 33, no. 2, pp. 64–68, Feb. 2000.

[166] *Robust Face Detection Using the Hausdorff Distance.*, ser. Lecture Notes in Computer Science, vol. 2091. Springer, 2001.

[167] C.-N. FAQ. Should i normalize / standardize / rescale the data? Online: ftp://ftp.sas.com/pub/neural/FAQ2.html#A_std. [Online]. Available: ftp://ftp.sas.com/pub/neural/FAQ2.html#A_std

[168] P. L. Bartlett, "For valid generalization the size of the weights is more important than the size of the network," in *Advances in Neural Information Processing Systems*, M. C. Mozer, M. I. Jordan, and T. Petsche, Eds., vol. 9. The MIT Press, 1997.

[169] D. Whitley, K. Mathias, and P. Fitzhorn, "Delta coding: An iterative search strategy for genetic algorithms,," in *Proceedings of the Fourth International Conference on Genetic Algorithms*, R. Belew and L. Booker, Eds. San Mateo, CA: Morgan Kaufman, 1991, pp. 77–84.

[170] J. Bruton. (2005) Intelligent Systems – EE551 Taught M.Eng. Course Notes. Dublin City University. [Online]. Available: http://www.eeng.dcu.ie/~brutonj

[171] M. Casas-Sanchez, J. Rizo-Morente, and C. J. Bleakley, "Power Consumption Characterisation of the Texas Instruments TMS320VC5510 DSP," in *Proc. International Workshop on Power and Timing Modelling, Optimization and Simulation (PATMOS)*, Leuven, Belgium, Sep. 20–23, 2005.

[172] ARM Integrator Compact Platform. ARM LTD. Cambridge, UK. [Online]. Available: http://www.arm.com/products/DevTools/IntegratorCP.html

[173] (2008) Blepo computer vision library. [Online]. Available: http://www.ces.clemson.edu/~stb/blepo/

[174] G. A. Ramirez and O. Fuentes, "Face detection using combinations of classifiers," in *Computer and Robot Vision, 2005. Proceedings. The 2nd Canadian Conference on*, May 2005, pp. 610–615.

[175] Y.-S. Huang, H. ying Cheng, P.-F. Cheng, and C.-Y. Tang, "Face detection with high precision based on radial-symmetry transform and eye-pair checking," in *Video and Signal Based Surveillance, 2006. AVSS '06. IEEE International Conference on*, Sydney, Australia, Nov. 2006, pp. 62–62.

[176] B. Froba and C. Kublbeck, "Robust face detection at video frame rate based on edge orientation features," in *Automatic Face and Gesture Recognition, 2002. Proceedings. Fifth IEEE International Conference on*, Washington, DC, May 2002, pp. 327–332.

[177] J. Wu and Z.-H. Zhou, "Efficient face candidates selector for face detection," *Elsevier Journal on Pattern Recognition*, vol. 36, no. 5, pp. 1175–1186, May 2003.

[178] A. F. Murray, D. del Corso, and L. Tarassenko, "Pulse-stream VLSI neural networks mixing analog and digital techniques," *IEEE Transactions on Neural Networks*, vol. 2, pp. 193–204, Mar. 1991.

[179] G. Cauwenberghs and M. Bayoumi, *Learning on Silicon - Adaptive VLSI Neural Systems*. Kluwer Academic, 1999.

[180] D. Zhang and S. K. Pal, *Neural Networks and Systolic Array Design*. New Jersey: World Scientific, 2002.

[181] U. Ruckert, "ULSI architectures for artificial neural networks," *IEEE Micro*, vol. 22, no. 3, pp. 10–19, May 2002.

[182] A. R. Omondi and J. C. Rajapakse, *FPGA Implementatoins of Neural Networks*, 1st ed. Dordrecht, Netherlands: Springer, 2006.

[183] S. Kung and J. Hwang, "Digital VLSI Architectures for Neural Networks," in *IEEE International Symposium on Circuits and Systems*, vol. 1, May 1989, pp. 445 – 448.

[184] Y. Kondo, Y. Koshiba, Y. Arima, M. Murasaki, T. Yamada, H. Amishiro, H. Mori, and K. Kyuma, "A 1.2 GFLOPS Neural Network Chip for High-Speed Neural Network Servers," *IEEE Journal of Solid-State Circuits*, vol. 31, no. 6, pp. 860–864, Jun. 1996.

[185] B. Denby, P. Garda, B. Granado, C. Kiesling, J. C. Prevotet, and A. Wassatsch, "Fast Triggering in High-Energy Physics Experiments Using Hardware Neural Networks," *IEEE Transactions on Neural Networks*, vol. 14, no. 5, pp. 1010–1027, Sep. 2003.

[186] S. Kung and J. Hwang, "A Unified Systolic Architecture for Artifical Neural Networks," *Journal of Parallel and Distributed Computing*, vol. 6, no. 2, pp. 358–387, Apr. 1989.

[187] C. Lehmann, M. Viredaz, and F. Blayo, "A Generic Systolic Array Building Block For Neural Networks with On-Chip Learning," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 4, pp. 400 – 407, May 1993.

[188] M. Viredaz and P. Ienne, "MANTRA I: a systolic neuro-computer," in *Proceedings of 1993 International Joint Conference on Neural Networks*, vol. 3, Nagoya, Japan, Oct. 1993, pp. 3054 – 3057.

[189] J.-S. Ker, Y.-H. Kuo, and B.-D. Liu;, "Design of a color reproduction neural network chip with on-chip learning capability," in *Proceedings., International Conference on Image Processing, 1996*, vol. 2, Sep. 1996, pp. 1023 – 1026.

[190] J. Cloutier, E. Cosatto, S. Pigeon, F. R. Boyer, and P. Y. Simard, "Vip: An fpga-based process for image processing and neural networks," in *Proceedings of IEEE MicroNeuro '96*, 1996, pp. 330–336.

[191] A. Schmid, Y. Leblebici, and D. Mlynek, "Hardware realization of a hamming neural network with on-chiplearning," in *Proceedings of the IEEE International Symposium on Circuits and Systems. ISCAS '98.*, vol. 3, May31, 1998, pp. 191–194.

[192] H. Amin, K. Curtis, and B. Hayes Gill, "Two-ring systolic array network for artificial neural networks," *IEE Proceedings Circuits, Devices and Systems*, vol. 146, pp. 225 – 230, Oct. 1999.

[193] J. L. Ayala, A. G. Lomena, M. L. Lopez-Vallejo, and A. F. Herrero, "Design of a Pipelined Hardware Architecture for Real-Time Neural Network Computations," in *IEEE International Midwest Symposium on Circuits and Systems*, Tulsa, Oklahoma, USA, August 4-7 2002.

[194] H. Kung, "Why systolic architectures?" *IEEE Computer*, vol. 15, pp. 37–46, Jan. 1982.

[195] G. G. Lendaris, R. M. Pap, R. E. Saeks, C. R. Thomas, and R. M. Akita, "Hardware Neural Network Implementation of Tracking System," in *Proceedings of IEEE Workshop on Neural Networks for Signal Processing*, Ermioni, Greece, Sep. 1994, pp. 451–460.

[196] M. A. Arbib, *The Handbook of Brain Theory and Neural Networks*, 2nd ed. Cambridge, MA: MIT Press, 2002.

[197] Szabo, T. and Feher, B. and Horvath, G., "Neural network implementation using distributed arithmetic," in *International Conference on Knowledge-Based Intelligent Electronic Systems*, vol. 3, Apr. 1998, pp. 510–518.

[198] F. Elguibaly, "A fast parallel multiplier-accumulator using the modified booth algorithm," *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing*, vol. 47, no. 9, pp. 902–908, Sep. 2000.

[199] Y. Liao and D. Roberts, "A high-performance and low-power 32-bit multiply-accumulate unit with single-instruction-multiple-data (simd) feature," *IEEE Journal of Solid-State Circuits*, vol. 37, pp. 926–931, Jul. 2002.

[200] D. Tan, A. Danysh, and M. Liebelt, "Multiple-precision fixed-point vector multiply-accumulator using shared segmentation," in *ARITH '03: Proceedings of the 16th IEEE Symposium on Computer Arithmetic (ARITH-16'03)*. Washington, DC, USA: IEEE Computer Society, 2003, p. 12.

[201] A. Fayed, W. Elgharbawy, and M. Bayoumi, "A data merging technique for high-speed low-power multiply accumulate units," in *IEEE International Conference on Acoustics, Speech, and Signal Processing, 2004*, vol. 5, May 2004, pp. 145–148.

[202] O. Kwon, K. Nowka, and J. Earl E. Swartzlander, "A 16-bit by 16-bit mac design using fast 5: 3 compressor cells," *J. VLSI Signal Process. Syst.*, vol. 31, no. 2, pp. 77–89, 2002.

[203] D. Myers and R. Hutchinson, "Efficient implementation of piecewise linear activation function for digital vlsi neural networks," in *Electronics Letters*, vol. 25, no. 24, Nov. 1989, pp. 1662–1663.

[204] C. Alippi and G. Storti-Gajani, "Simple approximation of sigmoid functions: Realistic design of digital vlsi neural networks," in *Proceedings of the IEEE Int'l Symp. Circuits and Systems*, Jun. 1991, pp. 1505–1508.

[205] H. Amin, K. Curtis, and B. Hayes Gill, "Piecewise linear approximation applied to nonlinear function of a neural network," *IEE Proceedings Circuits, Devices and Systems*, vol. 144, pp. 313 – 317, Dec. 1997.

[206] S. Vassiliadis, M. Zhang, and J. Delgado-Frias, "Elementary function generators for neural-network emulators," *IEEE Transactions on Neural Networks*, vol. 11, no. 6, pp. 1438 – 1449, Nov. 2000.

[207] H. Faiedh, Z. Gafsi, and K. Besbes, "Digital hardware implementation of sigmoid function and its derivative for artificial neural networks," in *Proceedings. of the 13th International Conference on Microelectronics*, Rabat, Morocco, Oct. 2001, pp. 189 – 192.

[208] K. Basterretxea, J. Tarela, and I. del Campo, "Approximation of sigmoid function and the derivative for hardware implementation of artificial neurons," *IEE Proceedings of Circuits, Devices and Systems*, vol. 151, no. 1, pp. 18–24, Feb. 2004.

[209] M. Tommiska, "Efficient digital implementation of the sigmoid function for reprogrammable logic," *IEE Proceedings Computers and Digital Techniques*, vol. 150, pp. 403 – 411, Nov. 2003.

[210] A. W. Savich, M. Moussa, and S. Areibi, "The impact of arithmetic representation on implementing MLP-BP on FPGAs: A study," *IEEE Transactions on Neural Networks*, vol. 18, no. 1, pp. 240–252, Jan. 2007.

[211] S. Draghici, "On the capabilities of neural networks using limited precision weights," *Neural Networks*, vol. 15, no. 3, pp. 395 – 414, Apr. 2002.

[212] J. Holt and J. Hwang, "Finite precision error analysis of neural network hardware implementations," *IEEE Transactions on Computers*, vol. 42, no. 3, pp. 280 – 291, Mar. 1993.

[213] H. Wust, K. Kasper, and H. Reininger, "Hybrid number representation for the FPGA-realization of a versatile neuro-processor," in *Euromicro Conference, 1998. Proceedings. 24th*, vol. 2, Vasteras, Aug. 1998, pp. 694–701.

[214] B. Gaines, *Stochastic Computing Systems*, *Advances in Information Systems Science*. Plenum Press New York, 1969.

[215] B. D. Brown and H. C. Card, "Stochastic Neural Computation I: Computational Elements," *IEEE Transactions on Neural Networks*, vol. 50, no. 9, pp. 891–905, Sep. 2001.

[216] D. Larkin, A. Kinane, V. Muresan, and N. O'Connor, "An Efficient Hardware Architecture for a Neural Network Activation Function Generator," in *International Symposium on Neural Networks*, Chengdu, China, May 2006.

[217] The MathWorks - MATLAB and Simulink for Technical Computing. Http://www.mathworks.com/.

[218] A. Omondi, *Computer Arithmetic Systems*. Prentice Hall, 1994.

[219] T. Schoenauer, S. Atasoy, N. Mehrtash, and H. Klar, "Neuropipe-chip: A digital neuroprocessor for spiking neural networks," *IEEE Transactions on Neural Networks*, vol. 13, no. 1, pp. 205–213, Jan. 2002.

[220] R. Gadea, J. Cerda, F. Ballester, and A. Macholi, "Artificial neural network implementation on a single FPGA of a pipelined on-line backpropagation," *2000. Proceedings. The 13th International Symposium on System Synthesis*, pp. 225–230, Sep. 2000.

[221] D. Lettnin, A. Braun, M. Bodgan, J. Gerlach, and W. Rosenstiel, "Synthesis of embedded systemc design: a case study of digital neural networks," *Design, Automation and Test in Europe Conference and Exhibition, 2004. Proceedings*, vol. 3, pp. 248–253, Feb. 2004.

[222] S. Popescu, "Hardware implementation of fast neural networks using CPLD," in *Neural Network Applications in Electrical Engineering, 2000. NEUREL 2000. Proceedings of the 5th Seminar on*, Belgrade, Sep. 2000, pp. 121–124.

[223] K. M. Yang, M. T. Sun, and L. Wu, "A family of VLSI designs for the motion compensation block-matchingalgorithm," *IEEE Transactions on Circuits and Systems*, vol. 36, no. 10, pp. 1317–1325, Oct. 1989.

[224] T. Komarek and P. Pirsch, "Array architectures for block matching algorithms," *IEEE Transactions on Circuits and Systems*, vol. 36, no. 10, pp. 1301–1308, Oct. 1989.

[225] C. H. Hsieh and T. P. Lin, "VLSI architecture for block-matching motion estimation algorithm," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 2, no. 2, pp. 169–175, Jun. 1992.

[226] E. Chan and S. Panchanathan, "Motion Estimation Architecture for Video Compression," *IEEE Transactions on Consumer Electronics*, vol. 39, no. 3, pp. 292–297, Aug. 1993.

[227] Y.-S. Jehng, L.-G. Chen, and T.-D. Chiueh, "An efficient and simple VLSI tree architecture for motionestimation algorithms," *Signal Processing, IEEE Transactions on [see also Acoustics, Speech, and Signal Processing, IEEE Transactions on]*, vol. 41, no. 2, pp. 889–900, Feb. 1993.

[228] P. N. Swamy, I. Chakrabarti, and D. Ghosh, "Architecture for motion estimation using the one-dimensional hierarchical search block-matching algorithm," in *Computers and Digital Techniques, IEE Proceedings* -, vol. 149, no. 5, Sep. 2002, pp. 229–239.

[229] H. Nakayama, T. Yoshitake, H. Komazaki, Y. Watanabe, H. Araki, K. Morioka, J. Li, L. Peilin, S. Lee, H. Kubosawa, and Y. Otobe, "A MPEG-4 Video LSI with an Error Resilient Codec Core Based on a Fast Motion Estimation Algorithm," in *Proc. of IEEE ISSCC*, vol. 2, Feb. 2002, p. 296.

[230] Y.-W. Huang, S.-Y. Chien, B.-Y. Hsieh, and L.-G. Chen, "Global elimination algorithm and architecture design for fast block matching motion estimation," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 14, no. 6, pp. 898–907, Jun. 2004.

[231] S. Y. Yap and J. V. McCanny, "A VLSI architecture for variable block size video motion estimation," *Circuits and Systems II: Express Briefs, IEEE Transactions on [see also Circuits and Systems II: Analog and Digital Signal Processing, IEEE Transactions on]*, vol. 51, no. 7, pp. 384–389, Jul. 2004.

[232] C.-Y. Kao and Y.-L. Lin, "An amba-compliant motion estimation for h.264 advanced video coding," in *Proceedings of 2004 International Conference on SOC Design, 2004*, 2004.

[233] L. Deng, W. Gao, M. Z. Hu, and Z. Z. Ji, "An efficient hardware impleemtnation for motion estimation of the avc standard," *IEEE Transactions on Consumer Electronics*, vol. 51, no. 4, pp. 1360–1366, Nov. 2005.

[234] C. Wei, M. Z. Gang, L. Z. Qiang, and Z. Yan, "VLSI architecture design for variable-size block motion estimation in MPEG-4 AVC/h.264," in *Circuits and Systems, 2004. Proceedings. The 2004 IEEE Asia-Pacific Conference on*, vol. 1, Dec. 2004, pp. 617–620.

[235] C. Wei and M. Z. Gang, "Reconfigurable VLSI architecture for VBSME in MPEG-4 AVC/h.264," in *ASIC, 2005. ASICON 2005. 6th International Conference On*, vol. 1, Oct. 2005, pp. 265–269.

[236] C. Wei and M. Z. Gang, "A novel VLSI architecture for VBSME in MPEG-4 AVC/h.264," in *Circuits and Systems, 2005. ISCAS 2005. IEEE International Symposium on*, May 2005, pp. 1794–1797.

[237] Y.-K. Lai and L.-G. Chen, "A data-interlacing architecture with two-dimensional data-reuse forfull-search block-matching algorithm," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 8, no. 2, pp. 124–127, Apr. 1998.

[238] J.-C. Tuan, T.-S. Chang, and C.-W. Jen, "On the data reuse and memory bandwidth analysis for full-searchblock-matching VLSI architecture," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 12, no. 1, pp. 61–72, Jan. 2002.

[239] Y. Song, Z. Liu, S. Goto, and T. Ikenaga, "A scalable vlsi architecture for variable block size integer motion estimation in h.264/avc," *IEICE Transactions Fundamentals*, vol. 89, no. 4, pp. 979–987, Apr. 2006.

[240] V. Do et al., "A Low-Power VLSI Architecture for Full-Search Block-Matching Motion Estimation," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 8, no. 14, pp. 393 – 398, Aug. 1998.

[241] L. Sousa and N. Roma, "Low-power array architectures for motion estimation," in *Multimedia Signal Processing, 1999 IEEE 3rd Workshop on*, Copenhagen, Denmark, 1999, pp. 679–684.

[242] K.-H. Lam and C.-Y. Tsui, "Low power 2-d array VLSI architecture for block matching motionestimation using computation suspension," in *Signal Processing Systems, 2000. SiPS 2000. 2000 IEEE Workshop on*, Lafayette, LA, USA, 2000, pp. 60–69.

[243] R. Richmond and D. Ha, "Low-power motion estimation block for low bit-rate wireless video," in *Proceedings of the International Symposium on Low Power Electronics and Design (ISLPED), CA, USA*, Aug. 2001, pp. 60–63.

[244] M. Takahashi, T. Nishikawa, M. Hamada, T. Takayanagi, H. Arakida, N. Machida, H. Yamamoto, T. Fujiyoshi, Y. Ohashi, O. Yamagishi, T. Samata, A. Asano, T. Terazawa, K. Ohmori, Y. Watanabe, H. Nakamura, S. Minami, T. Kuroda, and T. Furuyama, "A 60-MHz 240-mW MPEG-4 Videophone LSI with 16-Mb Embedded RAM," *IEEE Journal of Solid-State Circuits*, vol. 35, no. 11, pp. 1713–1721, Nov. 2000.

[245] S. Lopez, F. Tobajas, A. Villar, V. de Armas, J. F. Lopez, and R. Sarmiento, "Low cost efficient architecture for h.264 motion estimation," in *Circuits and Systems, 2005. ISCAS 2005. IEEE International Symposium on*, May 2005, pp. 412–415.

[246] Y.-K. Ko, H.-G. Kim, J.-W. Lee, Y.-R. Kim, H.-C. Oh, and S.-J. Ko, "New motion estimation algorithm based on bit-plane matching and its VLSI implementation," in *TENCON 99. Proceedings of the IEEE Region 10 Conference*, vol. 2, Cheju Island, Sep. 1999, pp. 848–851.

[247] A. Celebi, S. Erturk, and A. Tangel, "A vlsi implementation for fast all-boolean motion estimation based on pre-coded image planes matching," in *Conference on Electrical and Electronics Engineering (ELECO 2003)*, vol. 1, 2003, pp. 36–40.

[248] M. Mizuki and U. Desai and I. Masaki and A. Chandrakasan, "A binary block matching architecture with reduced power consumption and silicon area," in *Proc. IEEE ICASSP-96*, vol. 6, Atlanta, USA, 1996, pp. 3248–3251.

[249] B.Natarajan, V. Bhaskaran, and K. Konstantinides, "Low complexity block based motion estimation via one bit transform," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 7, no. 4, pp. 702–706, Aug. 1997.

[250] Feng, Lo, Mehrpour, and Karkowiak, "Adaptive block matching motion estimation algorithm using bit plane matching," in *IEEE Int Conf Image Processing, Washington D.C., USA,*, vol. NA, 1995, pp. 496–499.

[251] P. H. W. Wong and O. C. Au, "Modified one-bit transform for motion estimation," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 9, no. 7, pp. 1020–1024, Oct. 1999.

[252] X. Song, T. Chiang, X. Lee, and Y.-Q. Zhang, "New fast binary pyramid motion estimation for MPEG2 and HDTVencoding," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 10, no. 7, pp. 1015–1028, Oct. 2000.

[253] J. H. Luo, C. N. Wang, and T. Chiang, "A Novel All-Binary Motion Estimation (ABME) with Optimized Hardware Architectures," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 12, no. 8, pp. 700 – 712, Aug. 2002.

[254] D. Yu, S. K. Jang, and J. B. Ra, "Fast motion estimation for shape coding in mpeg-4," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 13, no. 4, pp. 358– 363, April 2003.

[255] K. Panusopone and X. Chen, "A fast motion estimation method for mpeg-4 artibitrarily shaped objects," in *Proc. of IEEE Int. Conf. Image Processing*, vol. 3, 2000, pp. 624–627.

[256] T.-H. Tsai and C.-P. Chen, "A fast binary motion estimation algorithm for mpeg-4 shape coding," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 14, no. 6, pp. 908 – 913, June 2004.

[257] H.-C. Chang, Y.-C. Chang, Y.-C. Wang, W.-M. Chao, and L.-G. Chen, "VLSI Architecture Design of MPEG-4 Shape Coding," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 12, no. 9, Sep. 2002.

[258] K.-B. Lee, H.-Y. Chin, N. Y.-C. Chang, H.-J. Hsu, and C.-W. Jen, "A Memory–Efficient Binary Motion Estimation Architecture for MPEG-4 Shape Coding," in *Proceedings of the European Conference on Circuit Theory and Design*, vol. 2, Sep. 2003, pp. 93–96.

[259] K.-B. Lee, H.-Y. Chin, N. Y.-C. Chang, H.-C. Hsu, and C.-W. Jen, "Optimal frame memory and data transfer scheme for mpeg-4 shape coding," *IEEE Transactions on Consumer Electronics*, vol. 50, no. 1, pp. 342–348, Feb. 2004.

[260] D. Larkin, V. Muresan, and N. O'Connor, "A Low Complexity Hardware Architecture for Motion Estimation," in *Proc. IEEE International Symposium on Circuits and Systems (ISCAS)*, Kos, Greece, May 21–24, 2006, pp. 2677–2680.