RUNTIME METHODS FOR ENERGY-EFFICIENT, IMAGE PROCESSING USING SIGNIFICANCE DRIVEN LEARNING

David Burke, C.Eng. MIET

A Thesis Submitted for the Degree of
Doctor of Philosophy at Newcastle University

School of Engineering
Faculty of Science, Agriculture and Engineering

July 2019

DECLARATION

I hereby declare that this thesis is my own work and effort and that it has not previously been submitted elsewhere for any award. Where other sources of information have been used, they have been acknowledged.

Newcastle upon Tyne April 2019

David Burke

CERTIFICATE OF APPROVAL

I confirm that, to the best of my knowledge, this thesis is from the student's own work and effort, and all other sources of information used have been acknowledged. This thesis has been submitted with my approval.

Dr. Rishad Shafik

Prof. Alex Yakovlev

To my wife Ann and daughter Kate

ACKNOWLEDGEMENTS

I would like to express my deep gratitude to my supervisors Dr. Rishad Shafik, Prof. Alex Yakovlev, for their wisdom and guidance through my PhD journey. They have always been a source of motivation.

I would like also to express my gratefulness and appreciation to my colleagues and friends in the School of Engineering, especially those in MicroSystems Research Group. We have worked together and discussed many topics over the years, and from them all have learned many things, I hope they continue to be successful with their research and future careers.

I would like to offer my special regards to all the staff of the School of Engineering in Newcastle university.

I need to thank my wonderful family for their continuous support and motivation throughout my PhD journey. My wife Ann who has always encouraged, motivated an supported me throughout my PhD and writing of this thesis. My daughter Kate for her encouragement and ended my many prevarications about taking up a research project when she said "Why don't you do a PhD Dad".

ABSTRACT

David Burke:
Runtime Methods for Energy-Efficient, Image Processing using Significance Driven Learning.

Image and Video processing applications are opening up a whole range of opportunities for processing at the "edge" or IoT applications as the demand for high accuracy processing high resolution images increases. However this comes with an increase in the quantity of data to be processed and stored, thereby causing a significant increase in the computational challenges. There is a growing interest in developing hardware systems that provide energy efficient solutions to this challenge. The challenges in Image Processing are unique because the increase in resolution, not only increases the data to be processed but also the amount of information detail scavenged from the data is also greatly increased. This thesis addresses the concept of extracting the significant image information to enable processing the data intelligently within a heterogeneous system.

We propose a unique way of defining image significance, based on what causes us to react when something "catches our eye", whether it be static or dynamic, whether it be in our central field of focus or our peripheral vision. This significance technique proves to be a relatively economical process in terms of energy and computational effort.

We investigate opportunities for further computational and energy efficiency that are available by elective use of heterogeneous system elements.

We utilise significance to adaptively select regions of interest for selective levels of processing dependent on their relative significance. We further demonstrate that exploiting the computational slack time released by this process, we can apply throttling of the processor speed to effect greater energy savings. This demonstrates a reduction in computational effort and energy efficiency a process that we term adaptive approximate computing.

We demonstrate that our approach reduces energy in a range of 50 to 75%, dependent on user quality demand, for a real-time performance requirement of 10 fps for a WQXGA image, when compared with the existing approach that is agnostic of significance. We further hypothesise that by use of heterogeneous elements that savings up to 90% could be achievable in both performance and energy when compared with running OpenCV on the CPU alone.

PUBLICATIONS

RELEVANT PUBLICATIONS

Journal publications:

1. **Dave Burke**; Dainius Jenkus; Issa Qiqieh; Rishad Shafik; Shidhartha Das; Alex Yakovlev, *Significance-Driven Adaptive Approximate Computing for Energy-Efficient Image Processing*, IEEE transactions on Computing. (to be submitted):

Conference publications:

1. **Dave Burke**; Rishad Shafik; Alex Yakovlev *Challenges and Opportunities in Research and Education of Heterogeneous Many-Core Applications*, 11th European Workshop on Microelectronics Education, EWME 2016 (2016) 1-6
2. **Dave Burke**; Dainius Jenkus; Issa Qiqieh; Rishad Shafik; Shidhartha Das; Alex Yakovlev, *Special Session Paper: Significance-Driven Adaptive Approximate Computing for Energy-Efficient Image Processing Applications*, International Conference on Hardware/-Software Codesign and System Synthesis (CODES+ISSS), October 2017, pp 1-2,
   doi: 10.1145/3125502.3125554,

Workshop and forum publications:

1. **Dave Burke**; Rishad Shafik, Adaptive Approximate Computing, Poster Session, ARM Research Summit 2017

PREVIOUS PUBLICATIONS AND PATENTS

Journal publications:

1. Roberts J; Ryley A; Jones D; **Burke D**; *Analysis of error-correction constraints in an optical disk*, Applied optics (1996) 35(20) 3915-3924

Patent Applications:

1. **Inventor David Burke**; Current Assignee Oracle America Inc; Original Assignee Sun Microsystems Inc; *GB9917511.9 26.07.1999 (GB2352548) Title: Method and apparatus for executing standard functions in a computer system*
2. **Inventor David Burke**; Current Assignee Oracle America Inc; Original Assignee Sun Microsystems Inc; *US6704816; 2004-03-09 Method and apparatus for executing standard functions in a computer system using a field programmable gate array*

CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACRONYMS

ACL	ARM Compute Library
AI	Artificial Intelligence
AVC	Advanced Video Coding
CAGR	Compound Annual Growth Rate
CBF	Cross Bilateral Filter
CCTV	Closed Circuit Television
CNN	Convolutional Neural Network
CPU	Central Processing Unit
CSV	Comma Separated Variables
CT	Clinical Tomography
DAS	Driver Assistance Systems
DCT	Discrete Cosine Transform
DCHWT	Discrete Cosine Harmonic Wavelet Transform
DUT	Device Under Test
DVFS	Dynamic Voltage and Frequency Scaling
DVS	Dynamic Voltage Scaling
FPGA	Field Programmable Gate Array
FPU	Floating Point Unit
GPU	Graphics Processing Unit
GPGPU	General Purpose Graphics Processing Unit
GUI	Graphical User Interface
HDD	Hard Disk Drive
HEVC	High Efficiency Video Coding
I2C	Inter Integrated Circuit
ILSVRC	ImageNet Large Scale Visual Recognition Challenge
IoT	Internet of Things

IIoT	Industrial Internet of Things
IP	intellectual property
JPEG	Joint Picture Experts Group
ML	Machine Learning
MOS	Mean Opinion Score
MRI	Magnetic Resonance Imaging
NDA	Non Disclosure Agreement
NPU	Neural Processing Unit
OpenXXX	OpenCV, OpenCL, OpenMP, OpenMV, ACL etc.
OpenMP	Open Multi Processor
OpenCL	Open Computing Language
OpenCV	Open Computer Vision
OpenAMP	Open Assymetric Multi Processing
OpenMV	Open Machine Vision
OPP	Operating Performance Points
PCB	Printed Circuit Board
PPQ	Power, Performance, Quality
ROIs	Regions of Interest
SDAAC	Significance Driven Adaptive Approximate Computing
SoC	System-on-Chip
SSD	Solid-state Disk Drive
SSIM	Structural Similarity
SMEs	Small to Medium Enterprises
UHD	Ultra High Definition
UIQI	Universal Image Quality Index

Part I

Thesis Chapters

1

INTRODUCTION

1.1 RATIONALE

Recent white papers and articles have indicated there may be a shift in interest, by Systems Architects, in the types of devices required for the Internet of Things (IoT). One such article by Altera (now Intel) [1] (Last checked 16 Sep 2019), questions the simplistic approach of the IoT and the cost of implementing individual single sensor devices. The article suggests an integrated approach and as an example suggests that, in a smart city application, the idea of a high definition camera, with video analytics, that can fulfil the function of a number of the single sensors, in this metropolitan scenario, while acting as a hub to gather data from existing single sensor devices. This vision produces a number of challenges to develop systems that can process significant volumes of high resolution image data, identify, extract and transmit the extra information content, while minimising energy usage, requiring highly efficient heterogeneous processing solutions.

Market forecasts show that Artificial Intelligence devices in IoT market are expected to grow from $5.1 billion to $16.2 billion in the period 2019 to 2024, at a Compound Annual Growth Rate (CAGR) of 26.0%, [2] (Last checked 16th Sept 2019). The major factors expected to drive the market are the need to efficiently process huge volumes of real-time data being generated from IoT devices while reducing maintenance costs and downtime.

Such IoT and other embedded devices show a broad range of applications including Image Processing at the Edge, such as:

- Smart-city applications for use in controlling traffic and pedestrian flow, smart Closed Circuit Television (CCTV) systems to record only significant events in the field of view, vehicle number plate recognition for admission to car parks.

- There are a number of safety critical Automotive applications, dozing driver recognition systems that can alert and awaken the driver before an accident occurs, Driver Assistance Systems (DAS) to ensure safety of vehicle passengers, pedestrians and other traffic.

- Financial security is a constant development field, which will always face persistent challenges from fraudsters and hackers, currently utilises fingerprint, voice and iris recognition. Facial recognition systems for such security purposes is an obvious future choice.

- Medical Imaging Processing has a multitude of applications addressing enhancement, recognition and display of various Magnetic Resonance Imaging (MRI), Ultrasound, optical, x-ray & Clinical Tomography (CT) and nuclear images. There is evidence that some of these techniques are migrating to portable personal health devices.
- Machine Vision, since the 2012 ImageNet Large Scale Visual Recognition Challenge (ILSVRC), is currently in a rapid growth development scenario across most of the above categories moving from a phase where facial recognition is almost de rigueur, to semantic segmentation where all individual objects in an image can be detected or recognised using pre-trained models. Typically this development is based on larger power computing systems for the learning and classification phase with a growth being seen in the detection phase on embedded or IoT systems, in some cases utilising dedicated Neural Processing Unit (NPU)s.

Such growth in Image Processing will create a significant demand for processing power and stored energy to power the processors. This creates a unique opportunity to create processes that can assist in improving the energy efficiency of these devices.

1.2 ENERGY EFFICIENCY

If we consider a simple example of a current 640 x 480 VGA graphics 30fps video stream which can be processed efficiently by a single embedded processor. Over a 24 hour period this would entail processing around 0.8 Terapixels which equates to around 3.2 Terabytes of storage. Growth of the image to a 4320 x 2432 Ultra High Definition (UHD) picture represents around 35 times growth in the number of pixels to be processed which not only increases the data and information content of the image with an associated increase in the transfer rate but also would necessitate a significant increase in the requirement of processors and their computational ability, by an equivalent amount, to process the frame. Such an increase requires heterogeneous resources that bring together special purpose processors, floating point devices, GPUs and perhaps FPGAs together. This allows the processor to manage the data collection and interact with the IOs with the GPUs utilised as special purpose processors, a growing number of FPGAs are now providing special software support in order to provide image processing in real time. This increase in computational power will also carry a penalty in the energy required for such computations. Realising the computational complexities and challenges of energy efficient computing there is a growing interest that requires novel approaches to enable identification of the significant features in an image, utilisation of approximate computation techniques and power management in order to increase energy efficiency.

1.3 RESEARCH QUESTIONS

In order to address the challenging topic of energy efficiency, the following research questions are relevant:

RQ1 Can we intelligently infer the significance of image blocks in Image Processing?

RQ2 Can we exploit the knowledge of significance to control computational complexity using application level approximation?

RQ3 Can we exploit a synergistic approach of hardware and software to maximise efficiency and reduce power consumption in heterogeneous systems in image processing workloads across different components?

1.4 CONTRIBUTIONS AND THESIS OVERVIEW

Current Image processing computation is effected by dividing the image into blocks to permit parallel computation. Each block will have a different level of information contained within but the same process is applied to every block.

This research has provided a software demonstrator that illustrates the use of approximate methods to derive the significance of the information contained in blocks of an image so that they can be selected for a level of processing complexity particular to their significance, addressing research question RQ1.

Further, utilising a two threshold level selection process based on user requested percentage levels, to effect a three level convolution kernel concept, with increasing accuracy kernels to be applied to the three increasing levels of incremental significance level blocks. This three level system could then be allocated to a particular processing engine, dependent on the on the complexity of the process. For example a high significance complex process may be executed on a GPU or FPGA, a mid significance process may be allocated to a DSP engine based process and low significance blocks allocated to the CPU, thereby addressing research questions RQ2 and RQ3.

The model also demonstrates these principles, via the Odroid XU4 platform, as to how Neon FPUs and GPU can potentially be used to tune power and performance and slack processing time can be utilised to further control power reduction by application of Dynamic Voltage and Frequency Scaling (DVFS) to the CPU or other available engines, addressing RQ3.

The major contributions of this thesis are summarized as follows:

In Chapter 3 "Significance in Image Processing": We address research question RQ1 by introducing and defining significance, based on a standard deviation derived from a local mean. We provide three methods for extracting significance. The first based on traditional standard deviation. A second more efficient method based on Absolute deviation, and a third based on approximate absolute deviation with the additional benefits of substantial savings in energy and performance.

In Chapter 5 "Significance Driven Adaptive Approximate Computing": We address RQ2 and demonstrate the efficiencies of utilising a novel methodology, utilising Image significance to identify a percentage based selection of areas of an image to be selected for tailored processing, the level of processing being dependent on the significance level of that particular area. This allows the opportunity to disqualify insignificant areas from any processing activity, thereby offering processing and energy saving opportunities. Further the reduction in processing time, particularly for video frames, yields slack time that can be explored to utilise DVFS on the CPU to provide further energy savings.

In Chapter 4 "CPU, GPU & Neon Energy and Performance Characterisation" along with Chapter 5: RQ3 is addressed. We demonstrate the potential performance increase and energy saving provided by use of Neon FPU and/or GPU in image processing. This facilitates further opportunities, for application of DVFS to both CPU and GPU in order to yield further efficiencies in energy and processing, a major advantage, especially for high resolution image processing.

This thesis is organized into six chapters:

Chapter 1 "Introduction": this chapter, introduced the aims and structure of this thesis.

Chapter 2 "Background and Literature Review" provides the motivations, objectives for the thesis and reviews current applicable literature.

In Chapter 3 "Significance in Image Processing": This chapter provides an overview of current methods of finding significance in an image and We then introduce further exploration of less computationally expensive and approximate deviation methods.

In Chapter 4 "CPU, GPU & Neon Energy and Performance Characterisation" demonstrates the potential performance increase and energy savings provided by use of Neon FPU and/or GPU in image processing.

Chapter 5 "Significance Driven Adaptive Approximate Computing": explores the use of a software model with image processing quality percentage level control knobs to dynamically adjust the Significance Threshold levels, and further utilise available slack time to allow application of DVFS to reduce energy consumption. This chapter also highlights roadblocks that hindered further development with the tools implemented during development and indicates future opportunities for further research and development of these topics.

In order to develop the underpinning background and also appreciate the work done in the existing literature, Chapter 2 will give a comprehensive coverage on approximate computing, hardware and software topics.

Chapter 6 "Conclusion and Further Opportunities " Summarises this thesis.

2

BACKGROUND AND LITERATURE REVIEW

This chapter will provide background knowledge relevant to this particular dissertation going into the topics of energy efficiency, approximate computation and the programming models available for image processing
It will also comprehensively review the literature or research work done to date, relevant to this particular work.
Section 2.1 will introduce the background to this research. Section 2.2 will outline the challenges arising from implementing image processing on increasing image definition. Section 2.3 will outline some of the recent advances in vision technology that are being implemented to deal with this scenario. Section 2.4 will review the literature associated with this wide field of research and the challenges of approximate computing across a number of computing domains. Section 2.5 presents a review of how achievements in four particular computing domains may offer cross-domain solutions. Section 2.6 summarises this chapter.

2.1 BACKGROUND

In Chapter 1 the Altera article [1], proposed an argument about new developments such that, instead of a huge bank of sensors, in say the intelligent city, that by using high definition (4k or 8k) video cameras to enable the capture of a variety of events that would otherwise need to use a large number of single sensor based devices. The camera could produce the same city management information in video format. This could avoid the large infrastructure building, maintenance and security costs. All this means that the total cost of ownership can be reduced while offering better safety, security and reliability of the smart city system. However the article raises the point that, if a 4k or 8k camera were used in such devices, then some sort of processing is required to extract the required data either by processing raw data in the cloud or by substantial processing power close to the sensors, termed ’edge’ processing.
Companies such as Cisco see the future IoT as consisting of application specific virtual servers operating outside the cloud of the data centres, in what they term as ’Fog computing’, utilizing lightweight Central Processing Unit (CPU)s supported by hardware accelerators, be they Floating Point Unit (FPU)s, Graphics Processing Unit (GPU)s, General Purpose Graphics Processing Unit (GPGPU)s, Neural Processing Unit (NPU)s, Field Programmable Gate Array (FPGA) or other evolving technology, appearing as a portable container utilizing a Java virtual

machine or OpenCL platform. Use of the cloud creates problems with latency, security and bandwidth as the internet in it's current form would struggle to cope with the streaming rates required.
As an example of a 4k capable vision system, Omnivision produce the OV24A10 image device, an active array of 5664 x 4248 pixels (24 Megapixels) operating at 30 frames per second (fps), the OV24A10 is currently their highest resolution image sensor currently available. The raw data rate this gives is around 580MPixels/second, bearing in mind that a pixel can be a number of bits in the range of 8 to 32 bits/pixel that will multiply the pixel rate. In the OV24A10 case the pixel data width is 10 bits, giving a RAW data rate of just under 6Gb/s. For reference, 2 Terabytes of disk storage, if it could handle the raw data rate, would hold 44 minutes worth of data. Such data transfer puts this application into the Volume and Velocity category of BIG data.

2.2 CHALLENGES

Image capture, processing and display of 4K and 8K bring a number of technical challenges in the transfer rates, storage of data and power efficiency. Table 2.1 shows the emerging challenge of data transfer rates and the quantity of Raw data produced over a 24 hour period compared with historic and future imaging and display resolutions. Columns 1 and 2 show the horizontal and vertical pixel counts of some common image formats increasing from 640 x 480 VGA up to 4320 x 2432 UHD and onward to 8k. Column 3 shows the image pixel count, the product of width and height. Columns 4, 5 and 6 show image products for 30 frames per second processing based on an 8 bit pixel value giving the data rate for column 4. Column 5 shows the total number of pixels that have each to be processed in a 24 hour period by existing image processing techniques, note that this does not include any convolution filter, MAC operations, activity which would increase this figure significantly. Column 6 shows a relative energy figure based purely on the known energy figure for the 640 x 480 process at 30fps, scaled up by the increase in pixels processed. Columns 7, 8 and 9 show similar calculations for a higher 60 frames per second operation.

It can be seen that for an order in magnitude increase in frame size that the exponential relationships yield around two orders of magnitude increase in Transfer rate, storage and energy requirements. This scenario creates new challenges in:

- The transmission and storage of data.
- Creating energy efficient systems to process the data.
- Designing viable approximate computing systems hardware and software.

- Trading high performance operation aspects to enable lower power consumption during critical power availability periods.

2.2.1 *Data Rates*

The significantly higher data rate associated with Ultra High Definition cameras, above 4k width, create new challenges at the Printed Circuit Board (PCB) level. Data transfer is usually achieved in these cases by the use of multiple lane, Gigabit rate, serial differential drivers for the PCB-camera interface and high bandwidth ethernet, WiFi or other interfaces between the embedded system and the cloud. Data Compression schemes such as Joint Picture Experts Group (JPEG), Advanced Video Coding (AVC) (also known as MPEG4 or H264) and High Efficiency Video Coding (HEVC) (H265) help mitigate these transmission demands by reducing data rates, but this is a particular use case of transmitting data between two points. JPEG compression is used for still images, typically yielding 10:1 compression ratio for a 20 Megapixel image, dependent on content. Tan [3] compares the data reduction performance, for video test data, between HEVC and AVC, measuring PSNR and a subjective audience test, Mean Opinion Score (MOS), to yield relative performance figures. As an example, the "Manege" UHD 3840 x 2160 60fps test video, would render a raw data rate of around 16 Terabits per second. The reduced data rates for a score of 35db PSNR and an acceptable level MOS score of 7, yields a data rate of around 16Mbps for AVC and 8Mbps for HEVC, representing a 1000:1 reduction for AVC and 2000:1 for HEVC, a significant saving.

2.2.2 *Data Storage*

Table 2.1 columns 5 and 8 show the 24 hour period data quantity as a total in TeraBytes for 30 and 60fps video, a pixel value is normally 24 bits held in a micro-system as a 4 Byte or 32 bit word, representing the amount of Raw data produced at the camera over this 24 hour period. Current top of the range hard disks are 6TB capacity and Solid State at 1TB. Clearly some form of data reduction is required to store the relevant information. The Closed Circuit Television (CCTV) Industry is currently heading towards H265 compression for 4k and above images, this still creates a computational challenge during processing with the data having to be decompressed before analysis of the full frame continuous historical data that has to be ”searched” for items of interest.

The previous subsection demonstrates the data rate savings of AVC and HEVC which translates directly into a compression figure which offers considerable data storage savings. If the transmitted/stored

Horizontal Width	Vertical Height	Pixel count	Raw 30fps Data rate	Pixels processed per 24 hours @30fps	24 hour Energy @30fps	Raw 60 fps Data rate	Pixels processed per 24 hours @60fps	24hour Energy @60fps
pixels	pixels		Mbits /sec	Tera pixels	Mjoules	Mbits /sec	Tera bytes	Mjoules
640	480	307200	737	3.2	0.2	1475	6.4	0.3
1024	768	786432	1887	8.2	0.4	3775	16.3	0.8
1600	900	1440000	3456	14.9	0.8	6912	29.9	1.5
2048	1152	2359296	5662	24.5	1.2	11325	48.9	2.5
4320	2432	10506240	25215	108.9	5.5	50430	217.9	11.1
8192	4608	37748736	90597	391.4	19.9	181194	782.8	39.8

Table 2.1: A selection of image sizes, data rates, processing and power requirements.

compressed data is to be subsequently remotely processed at a higher level other than simply being viewed, eg feature extraction or facial recognition, then decompression has to be implemented before processing of the reconstructed frame can take place thereby increasing the overall required computational effort. It needs to be borne in mind that such compression processes are lossy techniques with the resultant image's high frequency content being degraded. This thesis offers a solution to this post-compression, image processing phase by providing a methodology that can highlight significant areas of the image and limiting processing to those significant areas.

2.2.3 *Power Efficiency*

Considering the power requirements to drive embedded devices using a traditional micro-controller fitted to a PCB to capture the video from a camera and display it on a screen. Currently this could be done for the VGA size, 640 x 480, by use of a particular manufacturerś Evaluation board, containing a Cortex M7 and small video screen, for example, capable of running at up to 216MHz consumes around 1.5W.

If we now address driving 4K and 8K systems which in the case of one particular image array uses 3 x 1.25Gbit/sec lanes to transfer the image data. In order to handle these sort of frequencies we are near the practical limit of traditional processor based PCB design! Traces have to be kept very short and board layout require careful routing consideration. We would need to scale up the microsystem to an array of Arm Cortex-A series high-end application processors with around 2.0GHz operation, alongside a capable GPU to cope with such data. We can now estimate that we are looking at tens of Watts of power instead of single figure Watts.

In order to appreciate the extra energy requirements and processing time required by increasing image sizes, Chapter 4 will show the comparative results of increasing image size utilising a single processor and further comparing it with hardware accelerator solutions.

2.2.4 *Energy-Efficient Computing*

Heterogeneous computing systems have seen significant growth in various development platforms and software libraries. These software libraries are optimised and aimed at performance rather than energy efficiency. IoT devices and embedded systems along with Data centre power consumption is currently a highly motivational topic promoting interest in approximation methodology to enable energy saving and efficiency concepts in data processing. As an example, efficiency shortcomings have been identified in power conversion and distribution systems in data centres that show a result that 10% power saving at the server level can translate to 25% at the data centre input power, Brady [4], Beitelmal [5]. While this is a pure data centre problem, it reinforces the care that must be taken when considering embedded and IoT systems power demand and conversion, providing benefit from the cascading down of efficiencies and technologies utilised in the data centre.

2.2.5 *Approximate Computing System Design*

This research investigated the use of approximation methods to reduce computation which consequentially results in power reduction. Further power managment techniques are also investigated to offer further power reduction. This thesis introduces image significance based on standard deviation which can identify key areas of an image and explores variations of this statistical based method to enable minimisation of computation. The research then introduces an approximate form of image significance which allows a novel approach of multi level threshold classification of areas of the image to allow stepped accuracy convolution in areas of high significance down to least accurate, or no, convolution in areas of least significance. This scenario offers the further opportunities of application of Machine Learning to utilise the threshold levels as a control knob. A further control knob exists in exploration of utilising Dynamic Voltage and Frequency Scaling (DVFS) to reduce the remaining computational slack time available once each frame has been processed enabling a balancing of performance against power usage. This will be explained in Chapter 5

2.2.6 *High Speed Performance*

To effect such image processing with the current traditional approach, would normally require a Heterogeneous system implementation of high performance processor or preferably an array of such processors, using available software models and parallel programming techniques along with system devices such as DSP or GPU accelerators. Alternatively, previous experience has led to use of FPGAs to perform time critical data processing in a power critical application in conjunction with a processor to manage the data storage functions. The potential data rates that are evident in this project will require an integrated approach. It is believed that the functionality could benefit from the application of high end ARM processors coupled with Neon FPUs, GPU or possibly FPGA.

2.2.7 *IP issues when requiring component datasheets*

Typically when faced with a difficult operating challenge with some complex component or CPU it was usual to revert to the manufacturers data sheet in order to help resolve the problem, but that scenario is changing. Various component manufacturers are now very protective of their intellectual property (IP) and demand that potential users of their components, with a desire to build in their devices for some new product, will have to sign an Non Disclosure Agreement (NDA) before technical data such as data sheets or user manuals are released to the developers. Provision of an NDA to be signed may also come with a conditional restriction that there is an intention to source a considerable quantity of the components to build into the end product. As an example, Omnivision manufacture many of the computer vision cameras used in computer vision applications, they are now very protective of their IP such that they will not release a data sheet for their product until the potential user has given them full information on what the end product is to be. This presents a "no win" situation when the developer has to reveal his design plans, exposing the developers' potential IP, in order to be able to sign an NDA for the product being developed, a potential lockout situation.

Another such example is in the case of embedded processors and withholding release of IP by manufacturers as a barrier to rapid development by their competitors eg Samsung with its' Exynos 5422 octacore processor and graphics driver. The datasheet is only released under an NDA which is only offered to volume manufacturers.

2.3 Recent advances in imaging processing

Vision Technology is a rapidly advancing topic with continuous improvements in the camera technology along with the electronic devices and the software tools to process such images.

2.3.1 *Ultra High Definition Cameras*

There are a number of Semiconductor manufacturers with 4K Image arrays which are now appearing as manufactured CCTV cameras. 8K Image arrays are also now available but there is not much evidence of these appearing in camera form at the moment. These devices operate with multi lane GigaHerz+ Communication channels making traditional embedded system and PCB design more complex. An early opinion of this is that an FPGA System-on-Chip (SoC) design approach would yield a successful approach. With such high frequencies we are now nearing a time when current PCB design approach may need to change.

2.3.2 *Programming Models for Image Processing Applications*

There are a number of programming models that offer library functions for different aspects of system software design to support the development of various aspects of system operation. The OpenCV, OpenCL, OpenMP, OpenMV, ACL etc. (OpenXXX) series of toolboxes are Open Source libraries of software provided to enable growth in their particular field by enabling academics, professionals and entrepreneurs to implement projects and perform research by providing a logical and consistent C++ and/or Python interfaces to library functions for the particular library.

Open Multi Processor (OpenMP) is an API supporting parallel programming on multi-platform shared memory systems. It achieves this by multithreading where a master thread forks other runtime threads to be distributed via the runtime environment for concurrent running on available processors. OpenMP orignated in 1997 to support High Performance computing with C/C++ and Fortran. OpenMP is effective for multi platform Multiprocessor applications for high performance computing. At the moment we do not anticipate this to be an attractive proposition for this project.

Open Computing Language (OpenCL) originated in 2010 and allows parallel programming of diverse devices in heterogeneous platforms including CPUs, GPUs, GPGPUs, FPGAs and SIMD coprocessors eg Neon DSP engine. OpenCL is particularly useful where "embarassingly parallel" computation is normally performed in software. OpenCL offers the ability to perform intense software code in the hardware of GPUs or alternatively as hardware configured in an FPGA. Application of OpenCL in this thesis can only be reviewed once OpenCV functionality is fully understood. Research may show that some functionality may possibly not be able to be implemented directly with OpenCV and direct FPGA functionality programming may be required. Where these combinations of processor and accelerators are used significant higher performance can be gained from the use of GPUs, FPGAs or other devices as they are optimised to deal with data without the overhead of having to fetch the mixtures of program code and data or the overhead of fetch, decode, execute cycles, they also do not possess periodically redundant hardware items, registers, ALUs etc that are necessary for a processor to function. OpenCL aims at this type of application so that the application can be configured to run on processor only, processor/GPU or processor/FPGA combinations or indeed all three. OpenCL is still going through development stages but it is felt that it is now at the stage of requiring a real project for it to be applied to.

Open Computer Vision (OpenCV) library, started in 1999, driven by a number of trends such as mobile phone cameras, Internet search engines accessing image databases, the available power in modern computers and the maturity of vision algorithms which are now moving towards neural network processing. Such is the width of algorithms in OpenCV that the applications that can be generated are limited only by the imagination of humanity and their ability to translate their ideas into running code. OpenCV is effective at performing in depth Computer Vision Image analysis. It provides a useful set of algorithms for image analysis and will be a good starting point to investigate the methodology for detecting objects moving into and through the Field of View of a camera. AMD have instigated OpenCL acceleration of OpenCV since 2011 and OpenCV 3.x will include the Transparent API (TAPI) as sponsored by AMD and Intel. The OpenCL OpenCV TAPI interface might be some risk to the project as initial investigations give the impression that this interface is specifically for GPUs or GPGPU with no reference to FPGAs.

Open Assymetric Multi Processing (OpenAMP) started life in 2005 and is aimed at a broader range of heterogeneous and homogeneous processing architectures that include HPC cores Real-time cores, hardware accelerators and programmable logic.

Open Machine Vision (OpenMV) is a fairly recent development, 2013, targeted at supporting Machine Vision on low cost embedded Cortex-M processors running OpenCV modules under microPython. This is a milestone achievement to be able to run complex Vision Processing modules on a low power embedded processor.

To slightly complicate matters further, Arm withdrew the Mali GPU development kit, an early ARM centric OpenCL development platform, from public release and subsequently released the ARM Compute Library (ACL), aimed at mobile and embedded processors. This represents the embodiment of a selection of OpenCV functions and a selection of machine learning algorithms, implemented in OpenCL, in order to achieve best processing performance in computer vision and Machine Learning by utilising the Neon SIMD architecture and GPU Midgard and Bifrost architectures. While this Library offers significant performance improvement, by the application of DVFS to processors and GPU there is the possibility of selectable power vs performance efficiency. Chapter 4 will explore the relative performance and power efficiency gains of the use of Neon and GPUs in Image Processing over CPUs. Chapter 5 will discuss balancing of DVFS and computational performance to increase power efficiency.

Most of the above tools are qualified for use on a host system under the Ubuntu or Debian Linux operating system. It is possible that they could be run under another flavour of Linux but when problems arise, the linux community has to be referred to in order to find solutions. In such cases most answers point out that the user wasn't using Ubuntu or Debian flavours of Linux and that the user should switch to one of these. Requests to install Ubuntu on some organisations PCs result in a flat refusal and the offer may only possibly be for CentOS without super-user or sudo (administrator) access, which would seriously hamper development work when needing to download the required tools.

2.4 LITERATURE REVIEW

As previously shown in Table 2.1, emergent video recording cameras are moving to a 4k format, around 3840 x 2160 pixels with a frame resolution of 8.3Mpixels Ultra High Definition (UHD) format with a data transfer rate of around 500Mpixels per second at 60fps frame rate. These types of transfers are realised, typically as a 2.9Gb/s serial transfer rate after framing and encoding of the data stream. The data rates, data storage requirements and the processing power when compared with a 1k format image, all increase by around an order of magnitude, along with the infrastructure to support such recording. The move to 4k is driven by the extra resolution offering fine picture detail. Future trends indicate a move to even higher definition images with 8k cameras and beyond, again increasing the data rates, requirements for image processing and data storage. The 2016 roadmap issued by the Ethernet alliance shows a progression above 1Tb/s beyond 2020 as outlined by D'Ambrosia [6] . The most recent Global Ethernet traffic forecast from the Cisco expects data centre traffic to be 1.6 Zetabytes/-month (1 Zetabyte = 10^{21}) by 2021, up from 499 Exabytes/month in 2016, (1 Exabyte = 10^{18}), from the on-line Global- Data Center/ Cloud Traffic Forecast by Cisco [7] (Last checked 1 Oct 2019), select the "Cloud Traffic Highlights" link.

The insatiable demand for increased performance to process the extra information is now throttled by the limitation of the growth cycle and performance that were once a by product of the geometry stepping provided by Moore's law. There is a risk that this scenario could reduce capital funding for new areas of exploration. Thompson [8], discusses the pervasive effect of the slowing of Moore's law, the switch to specialised processing having an impact on overall market demands, which are reducing for processors and increasing for specialised computing, thereby accelerating this niche market. They define a fragmentation cycle, technology advance slows, fewer users adopt the slowing technology, thereby reducing available finance for innovation that advances technology.

The impact of these demands for higher resolution images creates a further challenge in data storage. While Solid-state Disk Drive (SSD) technology is rapidly replacing some magnetic Hard Disk Drive (HDD)s in data centres, the increase in storage density of SSDs are also impacted by Moores law. The evolution of magnetic recording is a source of continuous amazement, while Moore's law seems to have reached it's terminal growth step in device geometry, magnetic recording still has some way to go before the terminal recording density is reached. Over the years Wood predicted future recording densities [9], in 2000 of 1 $\mathrm{Tb/in^2}$, [10], in 2009 of 10 $\mathrm{Tb/in^2}$, these predictions took typically five to ten years to achieve as a finished product. In fact, for a sustained period prior to 2002, magnetic storage exceeded Moore's law in doubling aereal density every year rather than 18 months [11]. Further research is demonstrating that 1 $\mathrm{Tb/in^2}$ has been achieved in 2013, in the laboratory by Wu [12] indicating the onward strides in HDD magnetic recording.

The effects of such image developments will lead to significant growth in data centres to transfer, store and process data. This has the knock on effect of increased power demand by the data centres. An Information Week 2013 survey on Data centre power, average power is 8.5kW per rack with 27% of operators reporting >10KW per rack, Kurt [13]. In the drive for efficiency and reducing operating costs, the data centres are now demanding higher efficiency in equipment power consumption.

This is not only a problem for data centres. Considering the traditional current design of an 'intelligent' CCTV system utilising a number of cameras, typically up to 16 remote cameras per system, directly connected to the CCTV system box via coax or cat5 ethernet cable. Each camera channel raw data is image processed and if movement is detected, that channel is compressed and recorded to the disk. If such an image system were to be up-scaled from HD to UHD, a four times increase in pixels, the 3Gbit/s UHD camera output would be unlikely to be directly connected using traditional coax or Ethernet. In all probability the camera would require some form of processor so that the signal would need to be compressed before transmission. This further complicates the CCTV recording system as the input signal would need a processor to decompress the data stream, before image processing takes place, this probably requiring four times the number of processors, then the images would need to be compressed again before storage. This indicates a significant increase of processing power and an accompanying increase in power usage. The alternative, this research proposes, is to perform all the image processing at the camera, reduce the quantity of data, before compression and transmission to the CCTV system for storage.

2.5 APPROXIMATE COMPUTING IN SOFTWARE AND HARDWARE

Approximate Computing is a highly diverse subject field with a number of motivational drivers. The following review will consider and categorise their application into four main fields.

1. Solutions for Data Centre application that address potential power savings while maintaining or even increasing performance. This category in itself is a broad field of power conversion, distribution and storage, cooling techniques and equipment, server equipment and data storage equipment.

2. Solutions that require application specific hardware or tightly constrained architectures. Such solutions will generally appear as ASIC, SoC or possibly FPGA based products for integration into other computer solutions.

3. Solutions for application in existing hardware, be it desktop, portable or single board heterogeneous computers, basically computers that could be operated by attaching to a mains supply source.

4. Solutions for consumer, specialist or industrial embedded applications, generally mobile, that may have stricter power budget or performance limitations. This section includes mobiles, handheld, Internet of Things (IoT), Industrial Internet of Things (IIoT) and various types of energy scavenging or battery powered computer solutions.

2.5.1 *Data Centre Solutions*

As previously mentioned, data centres have a number of challenges with increasing data and demand to process that data, increasing the power consumed and lost in the various conversion and distribution chains which creates extra heat requiring additional cooling which escalates power consumption, a vicious energy consuming cycle.
Wang, [14] deals with Cloud applications of SoC clusters. Also deals with FPGA and cloud applications used in data transfer and compute intensive applications performance and proposes a performance model for both scenarios. Venkataramani [15] assesses the various approximate computing techniques across the computing stack from data centres down to embedded and IoT. One of the key principles being that approximate computing should yield disproportionate benefits, or large improvements in performance or energy with little or no impact

on output quality. Tatchell-Evans, [16] in a different approach outlines a novel method for cooling in data centres. Increasing the efficiency of the cooling air flow and reducing bypass flow, cools the equipment more efficiently and reduces equipment power consumption. Barroso [17] puts forward the case for energy-proportional computing. Essentially Googles' experience of server computation, power efficiency and the need to improve energy proportionality in data centres. Sidler [18] outlines an interesting application of a full segmentation offload, low latency, FPGA solution to the challenge of TCP and UDP frames normally handled by a software stack, reducing latency and power. This has been implemented as a solution by Intilop. Duben, [19] offers a preliminary study of the opportunities present in achieving energy savings through novel approaches such as inexactness, applied to large datasets.

The following papers are also aimed at servers in the data centre and demonstrate their applicability to constantly changing heavy load scenarios. These procedures require usage of a portion of the saved power so currently may not be suitable for low power embedded application but future developments could eventually witness these techniques migrate to the Existing Hardware category and possibly further to the Embedded category. Hoffman [20] proposes dynamic knobs for responsive power-aware computing. Essentially a power balancing tool for data centres, mixed strategies for servers with low or high idle power in order to optimise power performance and latency. Cheng [21] uses models at runtime, M@RT, to address assurance for self-adaptive systems, outlining the complexities of self-adapting models at run time, in various applications.

2.5.2 *Application Specific Solutions*

This section covers solutions that are realised as components that require or generate application specific hardware or tightly constrained architectures eg ASIC SoC or FPGA.

Shafique [22], provides an overall view of approximation techniques including software, architectural and hardware components with an emphasis on bridging the gap between these components for use in integration at system level and adaptive systems.

At the SoC component level, Mittal [23] discusses the potential of utilising domain wall memory for large, fast, low power memory in processor components. Venkatachalam [24] discusses fundamental power reduction techniques for microprocessor systems. An in depth exploration of the various circuit silicon level challenges to energy consumption and the methodology to optimise such usage. Tagliavini [25] offers an ASIC based, specialised implementation, utilising

Standard Cell Memory in place of SRAM in instances where supply voltage changes are used to reduce power and SRAM becomes unreliable, demonstrating application in the the PULP processor. Cyclops by Rahimi [26] details a low power and resolution, 352x288, image sensor to replace multiple environmental sensors. Ideal for large scale wireless sensor network deployment with low power, tightly constrained and architected for low power, but emphasises that high resolution images coupled with high speed processing requires a different platform. An energy efficient approximate multiplier based scheme is demonstrated by Qiqieh [27], which shows how underlying hardware processing can be simplified through progressive bit significance-driven logic compression. Building on the Qiqieh work, Esposito [28] introduces a low power approximate MAC unit, which could demonstrate a future lower power development path for implementation in conjunction with this project. Venkataramani [29], outlines scalable machine learning for image processing, an optimised machine learning process that reduces power consumption also [30], "Computing Efficiently", proposes a framework for judging whether approximation is giving “good enough” results.

Chippa discusses a sequence of processes in a number of papers for SoC application, [31] characterises inherent application resilience, the ability to produce acceptable results with approximate or inexact computation. Mainly aimed at Recognition Mining (RM) and search processes by a multi core RM Processor. [32] utilises dynamic effort scaling, SVM and K Means clustering, to modulate the effort, in hardware and software, that is expended in computing the results by dynamic management of the control knobs. [33] Brings together automatic resilience, scalable effort and dynamic effort scaling for approximate computing. These three techniques come together in [34] scalable-effort classifiers, a new approach to optimizing the energy efficiency of supervised machine-learning classifiers. Defines the structure of the RM Processor and the resultant energy savings for an acceptable reduced quality of results.

2.5.3 *Existing Hardware Solutions*

This section addresses techniques that can be implemented in existing general purpose computer systems hardware, be it desktops or laptops etc. but could also possibly be industrial applications that are powered by some type of external supply.

Van-den-Bergh, [35] introduces SEEDS, an algorithm that groups pixels belonging to the same object creating superpixels by using hill climbing optimization, reducing run time and power for an object detection system when compared with it’s predecessors. The algorithm seems to have been developed to run on an I7 target and the only reference to image size is 480 x 320. Kudia, [36] approximate computing can be employed for an emerging class of applications from various domains such as multimedia, machine learning and computer vision. The approximated output of such applications, even though not 100% numerically correct, is often either useful or the difference is unnoticeable to the end user. Appears to be a rather complicated, perhaps advanced, error checking network for approximation. Perhaps more suited to application as a debugging tool on an a new process that may require debugging for large errors! Rumba by Mishra, [37] introduces and explains use of iACT- Intel Approximate Computing Toolkit. It seems this could be a useful tool for checking approximation results but only for Intel processor based applications. Universal Image Quality Index (UIQI) by Wang, [38], an alternative quality measurement system to the MSE and PSNR measure, also [39] further develops Structural Similarity (SSIM) from UIQI measurement. This process utilises a function of (mean, standard deviation and correlation) between original and processed image to highlight structural similarity. Mesheye by Hengstler [40] uses multiple vision devices to detect motion with low resolution image devices, the derived parameters are used to process events from higher resolution cameras, as per the approach considered in this thesis in Section 2.5.7. This is quite an intensive processing solution, showing it's age with VGA camera as the main viewer and sub sampled kilopixel stereo cameras for feature extraction. Filtering techniques applied to sub-areas of an image as Saliency detection by Hou [41], this paper appears to generate results similar to those developed in this thesis but achieves it by temporal application of a Fourier Transformation which is claimed to be a "simple method". Language and compiler support for auto-tuning variable-accuracy algorithms by Ansel, [42] a method of achieving variable accuracy with autotuning during the compilation phase. A Reduce and Rank kernel by Raha, [43], performs a reduction operation (e.g., distance computation, dot product, L1-norm) between an input vector and each of a set of reference vectors, and ranks the reduction outputs to select the top reference vectors for the current input. Paraprox: Pattern-Based Approximation for Data Parallel Applications, Samadi [44] approximates kernel filtering with its stencil and partition algorithm by utilising a subset of the core pixel and its neighbours across the entire image.

2.5.4 *Embedded Hardware Solutions*

The demand for autonomous embedded systems requiring better performance but limited by power, is addressed here. This includes hardware utilising energy scavenging systems or battery powered systems which may include IoT and IIoT along with other specialised types of industrial embedded applications.

The dawn of more advanced image processing and neural networks in the embedded field has seen the evolution of small format boards based on high performance, low power, multiprocessor/GPU/DSP/NPU SoCs as used in modern mobile handsets to effect a Linux based solutions capable of demonstrating advanced applications eg. AlexNET on Odroid XU4 utilising pre-learned databases to effect in-field recognition or classification of objects.

In an ideal world, it would be possible to estimate energy requirements for an embedded system at the concept stage, rather than having to design, build, write the software, define energy requirements and re-design energy efficient software to drive it.
Energy-Efficient Design of Battery-Powered Embedded Systems by Simunic, [45], discusses some of the various approaches and indicates that, with a plethora of processor types and combinations available, such solutions are not a simple task or possibly cost effective solution for typical Small to Medium Enterprises (SMEs). At the embedded processor level, OpenMV, by Abdelkader, [46] an affordable development platform, utilises a dual embedded processor, Cortex-M7 with a lightweight version of Python, MicroPython2013, to implement a low-cost, low-power image processing platform. This implementation provides a comprehensive library containing some advanced image processing algorithms that can be easily implemented via Python. Nguyen, [47] discusses generation of MCU + FPGA essential, low power system to detect falls, uses Sobel filter to detect edges and operates at 9.3 fps at 666MHz. Rusci [48] utilises PULP processor to provide an integrated image device, albeit a small 128x64 image with a four core low power processor to extract relevant ROI image information and transmit correspondingly processed motion data, for onward processing and feature extraction.

Energy harvesting and power management for autonomous sensor nodes by Christmann [49] is a treatise on all the elements of an energy harvesting Wireless sensor network challenges. It gives a fair bit of detail on the various elements. Targeting a complex adaptive power storage scenario utilising super-caps and battery thus powering loads via super-caps and utilising excess power Vcap > Vmax to charge battery and using battery when Vcap< V min. Raghunathan, [50] outlines design considerations for solar energy harvesting, wireless, embedded systems. He gives an insight to the challenges of environmental energy harvesting, specifically photo-voltaic generation, conversion, storage, networking and power balancing/management. Rodriguez Arreola, [51] discusses approaches to transient computing in energy harvesting systems. He provides a methodology for survival in energy harvester driven systems and discusses check-pointing of variables that can be utilised to recover computation after a "brown out".

Real-Power Computing by Shafik [52] addresses the challenges presented for survivable computing in the embedded domain where the computational effort available is dictated by the amount of energy available. In addition, if energy storage becomes critically low then the system must be able to cope with suspension and recovery, survival mode, when energy storage recovers. Yakovlev [53] covers enabling survival instincts in electronic systems and discusses energy modulated computing, a challenge to develop hardware that can survive energy deficiency interludes. Grigorian, [54], provides a methodology to perform adaptive error analysis for approximate computing based on lightweight checks.

2.5.5 *Other Literature*

In order to investigate if any research may have been done in a similar area to this thesis, a search filters of "significance" and "approximate significance" along with combinations of "image" and "processing" was performed. The results were then manually further filtered to remove extraneous references. Search results of techniques and applications, not necessarily approximate, that use the term, or define "significance" yielded:-

Mohapatra, [55] utilises a DSP engine in 3D processing to detect significance by targeting the motion estimation element of image processing, by generating the sum of absolute differences between adjacent frames, and applying voltage scaling. Godtliebsen,[56] utilises a statistical significance gradient process to detect features in medical image analysis for clinical diagnosis where the high quality of results is preferential to lightweight or energy saving processing. Karakonstantis [57] highlights an operating scenario where accurate or approximate solutions to DCT/iDCT can be utilised, dependent on the level of resultant quality or power saving required, but mainly seems to be targeted at individually optimising task significance. Multi-focus and multi-spectral image fusion, based on pixel significance using discrete cosine harmonic wavelet transform, ShreyamshaKumar, [58] outlines generation of pixel significance using compute intensive wavelet transforms for medical imaging. While these papers individually provide a solution to expose significance in some form, the techniques used do not demonstrate a lightweight solution that can be aimed at embedded or IoT solutions such as that presented in this thesis.

A further search to investigate use of the term "Approximate significance", resulted in a number of papers where the two search words appear somewhere in the context, thereby requiring a further manual filtering to find any similarity to the use of the phrase in this thesis.

A survey of the plethora of techniques for approximate computing by Mittal, [59], presents references exposing what methodologies are available and how they perform. Xu, [60] offers an overall review of various research topic domains but highlights that further effort is required to bring approximation to the mainstream.

While the "Approximate significance" phrase did not appear in either of the above references, they are worth listing here, as between them they review a wide number of approximate computation techniques and systems and the sub word "significant" appears regularly within the text (significantly)! Other instances of papers where the two words appear independently follow, while they reveal a range of interesting approximation and other techniques however, none approach the methodology reported in this thesis.

Samadi [61] addresses an approximate optimisation process, SAGE, to more efficiently execute image processing via GPUs, but specifically aimed at CUDA kernels. A design methodology for Artificial Neural Networks utilising approximation techniques, Zhang, [62] reports approxANN, selecting significant neurons in artificial neural networks, approximates the computation and memory accesses of certain less critical neurons based on a criticality analysis to obtain energy efficiency under quality constraints. A temporal spectral technique for fast salient motion detection by Cui [63], utilises removal of redundant (static), features from video sequences and extracts the salient motion features by use of a Fourier transformation on the temporal frames. A mathematical exploration of significance in large datasets for analysis by neural networks, for use on large scale compute networks utilising Rough Set Theory by Bania, [64].

In all of this group of papers there was no incidence of the context "approximate significance" apart from the Bania paper where the two words appear next to each other in the Keywords after the Abstract.

2.5.6 *A Subjective Assessment of Technique Migration*

The relevant papers reviewed in this chapter, as previously mentioned, fall into four distinct areas which the associated techniques are aimed at, these being 1: Data Centres, 2: SoC applications, 3: Personal computing platforms (Desktop, laptop and mobile devices) and 4: Embedded platforms which includes IoT. Each of the areas have their own particular demands and disciplines. In Table: 2.2 through to Table: 2.5 we present a personal subjective assessment for each of these areas, 1 to 4, and indicate the possibility of migration of the techniques to other areas of application. Also to the right hand side of the application areas a scoring system has been used to indicate the effect of the technique on Power Reduction, Performance Enhancement and Quality of Service. A further three columns, based purely on personal experience and subjective scoring, have been added to indicate the potential effect on manufacturers of electronic and computing systems, that often have to deal with equipment lifetime issues of Reliability, Maintainability and Sustainability. The ultimate issue often arises in the form of component obsolescence in equipment that is demanded by the user to have a long lifetime, in excess of 20 years, and may often cause re-design issues in the equipment later life. The scoring for these six categories is as follows:

- 1 significant negative effect
- 2 negative effect
- 3 neutral effect
- 4 positive effect
- 5 significant positive effect

In order to explain this migration and scoring, if we take the Ethernet Roadmap, the first entry in Table: 2.2 which is included as an example. The roadmap will have a significant negative effect on power reduction but merely demonstrates how Ethernet speeds are expected to increase over the coming years. The roadmap is mainly concerned with supporting the Data Centre backbone communications, as increased consumer demand for on-line services and the consequential increase in equipment installations to deal with the demand increases the requirement for faster interconnection services. We have witnessed the migration of 1GB/s ethernet from the 1999 timeframe into today's home routers, so can probably expect further migration of evolving higher ethernet speeds to migrate down the path towards embedded systems. For the scoring, such increases in ethernet speed will add a significant incremental contribution to the power consumption of the next generation equipment along with other increases due to the ever increasing demand for performance, hence this technique has a significant negative effect on power reduction but a positive effect on performance enhancement. The Quality of service score can be expected to be positive as the Ethernet Alliance has a well ordered development path and takes great care in development of standards and approval of new techniques. The next three categories, Reliability, Maintainability and Sustainability, rarely appear in literature but are often some of the challenges that manufacturers and operators have to deal with and try to overcome in the long term, during development and in the subsequent lifetime of the equipment. Reliability or apparent reliability can be caused by many issues, lack of documentation, complexity of technique, inexperience of the user, lack of training, configuration issues or just bad design. Maintainability, the task of keeping the equipment in its correctly working state with relative ease. With complex equipment this is not always a simple task and may have to rely on built in debug facilities, or state logging as an aide to remote diagnosis. Sustainability, is the task of keeping equipment running through it's working life, the main culprit here being obsolescence of electronic component parts, as new techniques result in generation of higher performance near equivalents that aren't always a full functional swap out alternative, so some sort of compromise design activity may need to take place.

In some of the worlds largest data centres a vast amount of equipment is used to transfer, store and process data. The ability to provide a 24/7 service is critical so the backup power supply arrangements often result in a complex power conversion and distribution network leading to extra inefficiencies. The power usage is measured in the 100s of Megawatts range. In this case a small percentage saving in power consumption at the equipment level can result in a large financial benefit. Data Centres are therefore one of the prime motivators for reduction in energy usage whether it be by approximation methods or any other efficiency technique, Table: 2.2, they also provide a sandbox for carefully managed trials for tuning techniques for approximation methods.

SoC provides the opportunity to experiment with novel techniques, table: 2.3 and can provide dedicated hardware to explore Approximation and Quality of Service feedback statistics along with DVFS to minimise energy consumption while providing reasonable performance. These SoC devices once proven to be functionally capable, may then migrate to inclusion in Data Centre equipment designs or to the Personal Computing level.

Personal Computing includes items that can be attached to some form of main power supply, eg desktops, or battery operated items that can be periodically recharged from a main power supply, eg laptops, tablets, mobile phones and possibly embedded development boards used to explore new techniques for embedded systems. Table: 2.4 demonstrates that migration from this group is more likely to move down to embedded applications especially now that some semiconductor manufacturers are seen to be releasing the SoC devices that have been used in mobile phones, for use in the embedded market.

Embedded includes battery operated or power scavenging devices where low power is an absolute necessity. Usually new embedded techniques are so specialised that they are unlikely to migrate in an upward direction to the other three categories, Table: 2.5 but this category will benefit from techniques developed in the three other fields but it is felt that there needs to be balancing support of error checking and power demands or alternatively, waiting for the techniques to mature and run without error checking.

Many Hardware and software techniques, (including OpenCV), are aimed at achieving best performance and don't consider energy. Typical of this are applications in the medical and biological field and image database retrieval where accuracy of results are far more important than saving computation effort or power consumption. Consequently when searching for literature using "adaptive" and "filter" in the hope of finding some approximation references, an overwhelming return illustrating a very active research field especially

				Applicable area 1-Data centre 2-SoC 3-Personal comp 4-Embedded									
Title	Paper ref.	Author	Content notes	Prime area	Potential Migration to			Power reduction	Performance enhancement	Quality of Service	Reliability	Maintainabilty	Sustainability
2016 Ethernet Roadmap	[6]	D'Ambrosia	Ethernet roadmap	1	2	3	4	1	4	4	3	3	2
Computing approximately, and efficiently	[15]	Venkataramani	Assesses approximate techniques across the stack	1	2	3	4	4	4	3	3	3	3
The Case for Energy-Proportional Computing	[17]	Barroso	Google, power cf mobile. DVFS on servers	1	3	4		5	5	5	4	4	4
Preliminary Study: Inexact General Purpose Processors for High-Performance and Big-Data Applications	[19]	Duben	addresses processor, datapath and memory	1	3			5	5	5	3	3	3
An energy-efficient system on a chip platform for cloud applications	[14]	Wang	SoC cluster clouds	1	3			4	5	5	4	4	3
The feasibility of magnetic recording at 1 Terabit per square inch	[9]	R Wood	1Tb/in	1	3			3	5	4	3	3	3
The feasibility of magnetic recording at 10 terabits per square inch on conventional media	[10]	R Wood	10Tb/in	1	3			3	5	4	3	3	2
Recording technologies for terabit per square inch systems	[11]	R Wood	areal doubling	1	3			3	5	4	3	3	3
HAMR areal density demonstration of 1+ Tbpsi on spinstand	[12]	Wu	1Tb/in demo	1	3			3	5	4	3	3	3
Models at runtime for self adaptive systems	[21]	Cheng	M@RT software	1				4	4	5	5	5	4
Dynamic knobs for power aware computing balancing power	[20]	Hoffman	Power balancing in data centre	1				5	5	5	4	4	4
Data Centre Decision Time	[13]	Kurt	Data centre power	1				4	3	3	3	3	3
Low-latency TCP/IP stack for data center applications	[18]	Sidler	Optimise TCP/IP stack on FPGA	1				5	5	5	4	4	3
Investigation of bypass air in data centres, impact on power consumption	[16]	Tatchell-Evans	efficient cooling	1				5	5	3	4	4	3

Table 2.2: Data centre based techniques.

				Applicable area 1-Data centre 2-SoC 3-Personal comp 4-Embedded									
Title	Paper ref.	Author	Content notes	Prime area	Potential Migration to			Power reduction	Performance enhancement	Quality of Service	Reliability	Maintainabilty	Sustainability
Low-power approximate MAC unit	[28]	Esposito	Approximate MAC that could enhance this thesis	2	1	3	4	5	5	5	4	4	3
Approximate computing and the quest for computing efficiency	[30]	Venkataramani	judging framework	1	2	3	4	4	4	4	4	4	3
Scalable-effort classifiers for energy-efficient machine learning	[29]	Venkataramani	ML effort classification	1	2	3	4	4	4	3	3	3	3
Analysis of inherent application resilience for approximate computing	[31]	Chippa	Application resilience in datamining and search apps	2	1	3		5	5	4	4	4	4
Managing the Quality vs. Efficiency Trade-off Using Dynamic Effort Scaling	[32]	Chippa	DES approx to suit data/app	2	1	3		5	5	4	5	5	4
Approximate computing: An integrated hardware approach	[33]	Chippa	feedback control for first two	2	1	3		5	5	4	5	5	4
Scalable effort hardware design	[34]	Chippa	Brings all 3 together	2	1	3		5	5	5	5	5	5
Techniques for Architecting Processor Components Using Domain-Wall Memory	[23]	Mittal	Lowering RAM Power	2	1	3		5	4	4	5	5	3
Significance driven logic compression multipliers	[27]	Qiqieh	SDLC	2	1	3		5	5	4	3	3	3
Cyclops: In Situ Image Sensing and Interpretation in Wireless Sensor Networks	[26]	Rahimi	Low power embedded, argues that HD will require different approach	2	4			5	5	4	4	3	3
Cross-layer approximate computing: from logic to architectures	[22]	Shafique	survey for SoC applications	2				5	5	5	3	3	3
Always-on motion detection with application-level error control on a near-threshold approximate computing platform	[25]	Tagliavini	SRAM Power saving	2				5	5	5	5	3	3
Power reduction techniques for microprocessor systems	[24]	Venkatachalam	In depth survey of energy drain in processors	2				5	5	4	5	3	3

Table 2.3: SoC based techniques.

				Applicable area 1-Data centre 2-SoC 3-Personal comp 4-Embedded								
Title	Paper ref.	Author	Content notes	Prime area	Potential Migration to		Power reduction	Performance enhancement	Quality of Service	Reliability	Maintainabilty	Sustainability
iACT: A Software-Hardware Framework for Understanding the Scope of Approximate Computing	[37]	Mishra	Intel's Approximate computing toolkit	3	1	4	4	5	5	5	4	3
Language and compiler support for auto-tuning variable-accuracy algorithms	[42]	Ansel	variable accuracy at compilation	3	4		3	5	5	3	3	2
MeshEye: A Hybrid-Resolution Smart Camera Mote for Applications in Distributed Intelligent Surveillance	[40]	Hengstler	Utilises low res low power devices to detect movement then process higher res camera output	3	4		5	5	5	4	4	3
Saliency Detection: A Spectral Residual Approach	[41]	Hou	Uses Fourier transform fortemporal differences	3	4		5	5	5	5	4	3
Quality configurable reduce-and-rank for energy efficient approximate computing	[43]	Raha	Identify areas that may only require approximate computation	3	4		5	5	5	5	4	3
Paraprox: Pattern-Based Approximation for Data Parallel Applications	[61]	Samadi	Utilises six patterns to identify & process area subsets approximately in GPUs	3	4		4	5	5	5	3	3
A Universal Image Quality Index	[38]	Wang	Universal Image Quality Index	3	4		4	5	5	5	4	3
Image quality assessment: From error visibility to structural similarity	[39]	Wang	SSIM derived from UIQI	3	4		4	5	5	5	4	3
Rumba An Online Quality Management System for Approximate Computing	[36]	Khudia	detect approximation errors and correct accurately	3			4	5	5	5	3	3
Superpixel Extracted Via Energy-Driven Sampling	[35]	Van-den-Bergh	SEEDS, compromise for power and efficiency	3			5	5	4	4	3	3

Table 2.4: Personal compute platform techniques.

				Applicable area 1-Data centre 2-SoC 3-Personal comp 4-Embedded							
Title	Paper ref.	Author	Content notes	Prime area	Potential Migration to	Power reduction	Performance enhancement	Quality of Service	Reliability	Maintainabilty	Sustainability
Openmv: a Python Powered, Extensible Machine Vision Camera	[46]	Abdelkader	OpenMV low-cost low-power embedded vision micropython platform	4	3	5	5	5	5	5	4
Design Considerations for Solar Energy Harvesting Wireless Embedded Systems	[50]	Raghunathan	Challenges of environmental energy harvesting, conversion, storage, balancing	4	3	5	5	5	5	4	3
Energy harvesting and power management for autonomous sensor nodes	[49]	Christmann	Reviews various energy harvesting and management architectures	4		5	5	5	5	5	3
Dynamically adaptive and reliable approximate computing using light-weight error analysis	[54]	Grigorian	application specific error control using LWC	4		4	5	5	5	5	4
Low power architecture exploration for standalone fall detection system based on computer vision	[47]	Nguyen	low power MCU + FPGA fall detection system	4		5	4	5	5	5	3
Approaches to Transient Computing for Energy Harvesting Systems	[51]	Rodriguez Arreola	Survival strategy, checkpointing variables for recovery	4		5	5	5	5	4	4
An Event-Driven Ultra-Low-Power Smart Visual Sensor	[48]	Rusci	Uses 4 core Cortex cluster to extract event driven feature info to forward to cloud based system	4		5	5	5	4	4	3
Real-Power Computing	[52]	Shafik	surviveable embedded computing	4		5	5	5	5	5	4
Energy-Efficient Design of Battery-Powered Embedded Systems	[45]	Simunic	With available uP and approaches, energy efficient design is not a simple task	4		5	5	5	5	5	4
Enabling Survival Instincts in Electronic Systems: An Energy Perspective	[65]	Yakovlev	A challenge to develop hardware to survive energy deficient interludes	4		5	5	5	5	5	4

Table 2.5: Embedded target techniques.

in attempts to improve filtering techniques in the biological, medical and "big data" fields. As an example, Raghuwanshi [66] Proposes a methodology of Image identification signature for Database retrieval. Divides the image into sub areas before low pass filtering, tiling with tetrolets, high pass filtering, then applies energy and standard deviation characterisation for feature extraction to create a signature. Further developments of this concept by Raghuwanshi [67] enhances the previously cited work to reduce processing time and adds a feed forward system to calculate image indexes on texture, edge and colour. This technique appears to be computationally expensive which is to be typically expected in database applications.

2.5.7 *Identified Research Problems*

From the literature review and the four system categories identified, of data centre down to embedded applications, there are a myriad of system and hardware architectures, operating systems, software development platforms, producing a broad range of homogeneous, heterogeneous or even single micro-controller solutions each with their individual challenges. There is no "one size fits all" solution. Conflicting design and implementation trade-offs constitute one of the major challenges of many-core applications. As different applications have varied performance, power, quality, reliability, maintainability and sustainability requirements, meeting these demands for each domain of application is difficult with a generic design and implementation flow, Shafik [68] and Yakovlev [65].

Historical methods to tackle the increase in data rates and processing power have been to increase the processor operating frequency, effectively made achievable by the die shrink in the manufacturing processes. Most of the performance increases have topped out at around the 3GHz operating frequency and the method now used to handle the increase in processing power has been to increase the number of processors using multiprocessor techniques. This scenario has increased the amount of power required to not only drive the extra processors but also the air conditioning equipment in data centres to keep the equipment cool. Further recent developments in the Data Centres have seen FPGAs used as accelerators for specific tasks such as Ethernet, Universal Data Packet (UDP) processing by Sidler [18] and Intilop [69] to enable real time data extraction and encoding. This problem arises when sending files, > 4k size, across the internet. Files are broken down into a number of TCP/IP packets of Maximum Transmission Unit MTU size which is typically 1536 bytes which also includes the packet header and a checksum at the end of the packet. The packetisation is a straightforward task at the transmission end but at the receive end a number of issues arise. Each packet, on arrival will generate an interrupt for the processor to deal with the packet. The packets won't necessarily be in the same sending order or may even result in simultaneous arrival of packets due to Internet routing, thereby adding extra re-ordering or dropped packet challenges to the receiving end. A solution to this process was realised in FPGA which reduced the processing latency from 10's of microseconds down to around 100nS. Such implementations have shown a lower loading of CPU processes and a reduction in dropped packets and significant overall savings in power consumption.

OpenCV has demonstrated its usefulness in identifying features in video such as moving objects and recognising features. OpenCV algorithms have been developed with the aid of much higher definition still photography (>20Mp) where processing time is not a critical issue, so is able to process UHD video frames, but for video, the allowable processing time is dependent on the frame rate to enable video frame rate capture. The challenge for this research is to identify the relevant area of data to be more efficiently extracted within defined Regions of Interest (ROIs) in the scene.

A further problem to be addressed is, for instance, the amount of processing required on a UHD frame to detect significant events is a large task because of the quantity of pixels to be processed. Could it be possible to use either a reduced resolution image or a lower resolution camera, viewing the same scene as the UHD camera, suitably calibrated, to detect movement and define the area to be extracted by the UHD device thereby offering the potential of reducing the frame processing time? Alternatively would it be possible to utilise a mask to select a subset pattern of the UHD pixels and utilise the resultant smaller subset of image pixels to detect significant events?

2.5.8 *Research Challenges*

The development of heterogeneous systems is a challenging area and involves a steep learning curve in:

- Gaining new knowledge in heterogeneous architectures combined with high bandwidth interfaces.
- Investigation of novel techniques to improve performance without demanding high clock rates and reducing overall power consumption.
- Gaining familiarity with operating systems, Linux and Yocto.
- Understanding and utilising high levels of software integration.
- Most of the software library tools have not yet reached full maturity, they are still in development stage and as such present several challenging issues that require granting the user some level of administrator access to the operating system.

- Adopting use of emergent library facilities OpenCV, OpenCL, OpenMP.
- Gaining awareness of security implementation issues.
- Exploring operating system management methods and power gating to reduce power consumption.

2.6 SUMMARY

Approximate computing is a very active and dynamic subject area currently. Latest developments demonstrate migration of complex developments from initial concepts in SoC or FPGA, down the chain to data centres, to desktop or mobile applications then onwards to embedded systems. This has recently been witnessed in the Computer Vision and Machine learning areas where techniques, developed in software and realised in hardware components, are migrating down the chain from data centres towards portable then onward to embedded instantiations.

3 SIGNIFICANCE IN IMAGE PROCESSING

In this chapter a key methodology for extracting image significance is described based on a localised standard deviation. We then go on to describe how the computational effort can be reduced by exploring absolute deviation, pyramid image reduction with standard and absolute deviation, before moving onto approximate absolute deviation which produces an economical method for deriving image significance values. This work allows a definition of Image significance to be created. Section 3.1 Introduces the topic and the reason for the research. Section 3.2 Outlines the motivation for such research. Section 3.3 explains the rationale and the contributions we make to the subject of significance in image processing. In Section 3.4 we define image significance and the development of the processes used to exploit image significance, identifying three suitable candidates from five different techniques, explained in Sections 3.5 through 3.9. Section 3.10 illustrates the comparative execution time results for these different approaches. In Section 3.11 we briefly outline a case study that proved the viability of an adaptive approximate significance model concept for further progress. Section 3.12 outlines the proposed adaptive approximate computing approach underpinning the definition of significance and the proposed path for a transition to adaptive approximate computing. Section 3.13 explains the control variables to achieve the adaptive approach. Finally Section 3.14 Introduces the next phase of research, Significance Driven Adaptive Approximate Computing.

3.1 INTRODUCTION

Image processing applications, which include acquisition, processing and analysis of real-world digital images, are increasingly being employed in myriad of embedded and ubiquitous systems. These applications have two major challenges posed by their conflicting requirements of performance and energy efficiency. Firstly, with continued advancement of camera and sensing technologies, there is a persistent demand for higher resolution of the captured frames (i.e. images) that require decoding at real-time, Beckett [70]. As such, the volume of data to be processed over a given time is increasing rapidly (see Table. 2.1). Secondly, processing this ever-increasing volume of data requires more processing power. This has made achieving energy efficiency highly challenging, even when the underlying hardware/ software stack is by-design low-power.

3.2 BACKGROUND AND RELEVANT WORK

Table. 2.1, in Chapter 2, shows typical data rate, storage and energy requirements for current and future image resolutions. The storage requirements and energy consumption results were generated from simple experiments on an Odroid XU4 platform at 2.0GHz operating frequency. As can be seen, the volume of raw image data that needs processing over a 24 hour period increases dramatically with increasing image pixel dimensions. Over the years, significant research works have been carried out to address the energy efficiency of real-time image processing applications. Approximate computing has recently emerged as a promising approach, which leverages the intrinsic resilience of these applications to imprecision, Han & Mittal [71, 59]. Energy efficiency is achieved through replacing the compute-intensive hardware/software routines by low-complexity ones, which essentially allow the application to run faster at lower energy consumption, Duben [19]. However, the quality of processing is compromised to a point, which is deemed acceptable using subjective or objective metrics.

Existing works in approximate computing in the domain of image processing consist of various methods to determine significant areas of an image in order to target computational effort to those particular areas. Godtliebsen [56] addresses significance of features in noisy images by analysing the gradients at each pixel and generating significant streamlines to find significance peaks. A different approach is taken by Mohapatra [55], by creating a significance driven motion estimator using temporal difference on video streams. Vassiliadis [72] utilises a program model and runtime process to extract a Discrete Cosine Transform (DCT) based task significance to allocate lower significance blocks for more approximate levels of processing and thereby increase energy efficiency in approximate computations. Superpixels are used by Van-den-Bergh [35] to segment the image by grouping pixels that are part of the same object by using hill climbing optimization with an iterative refinement in a process entitled SEEDS. Image fusion is a challenging topic where composite images are required for enhanced image processing by integration of images from different types of camera, eg Infra-Red and visible spectrum. ShreyamshaKumar [58, 73] generates a pixel significance based on Discrete Cosine Harmonic Wavelet Transform (DCHWT) for multi-focal multi-spectral images in the first instance and a version based on a Cross Bilateral Filter (CBF) in the second instance.

This chapter addresses significance, which we describe as "what catches your eye". It transpires that, in the literature, there is an alternative equivalent nomenclature for significance, called saliency, references to which, we include here. The Learning OpenCV 3 manual by Kaehler [74] defines the saliency as "Where humans would look in a scene". The following papers all use the term "saliency". Viola uses this term in [75] to achieve object detection and in [76] for real-time facial detection, these techniques use the OpenCV Integral() function for calculation of standard deviation which we will explore in more detail in Section 3.5. Hou [41] utilises a log Fourier spectrum to facilitate object detection, in a process specified for use in Machine Learning and is computationally intensive. A temporal spectral technique for fast salient motion detection by Cui [63], utilises removal of redundant (static), features from video sequences and extracts the salient motion features by use of a Fourier Transformation on the temporal frames.

3.3 RATIONALE AND CONTRIBUTIONS

Images typically consist of areas where the contrast between colors define the artefacts and features of the image more than those without any contrast, Preston [77]. In other words, areas where the raw data variation is higher are of more informational value than others. It can be postulated that these informational values, i.e. significance, can be used to modulate the computation efforts with the aim of achieving energy minimization, while also retaining the best possible quality of images. In this project, we define this as significance-driven adaptive approximate computing. The main premise is to adopt higher-precision computing for image areas that are more significant, and conversely allocate low-precision and low-complexity computing for areas that are deemed less significant. In a parallel image processing system, this will then lead to a problem of appropriate hardware/ software allocation, coupled with DVFS decisions to achieve quality-aware energy reductions against real-time performance requirements. This is a departure from the existing approaches that are agnostic of data significance within images and demonstrate clear advantages of the proposed approach.

In this work, the following specific ***contributions*** are claimed:

1. For the first time the concept of significance in the context of image processing applications is defined.

2. We present three individual methods of extracting significance. Standard Deviation and Absolute Deviation based on a local area mean and a more computational and energy efficient Approximate Absolute deviation based on a mean of four adjacent work-areas, these work areas being a choice of 4x4, 8x8 or 16x16 size.

3. A parallel image processing approach using significance driven approximate computing is proposed. Core to this approach is hardware/software resource allocation, coupled with DVFS con-

trols to optimize the energy and quality trade-offs with real-time performance requirements.

4. A further proposal is an adaptive machine learning approach that is constrained by an energy budget along with a quality budget to act as control inputs to the significance level thresholding, thereby modulating the resultant precision based computation.

5. Alongside this work, but not reported here, two concrete case studies were performed: an application-specific hardware-based adaptive approximate image filter and a software-based variable-kernel based parallel convolution filter running on an Odroid XU-4 platform, demonstrating advantages over the existing approaches.

3.4 DEFINING IMAGE SIGNIFICANCE

Significance in the context of an image, is defined as areas where the pixel value deviation is significantly different to a local mean, such that, it exposes information features arising from changes in visual effects and perception, Godtliebsen [56]. The research originated by investigating, by means of a software demonstrator, based on OpenCV version3.3 to determine if significance in still images can be estimated through parallel inference of mean and standard deviation per image block. The calculation of mean and standard deviation was based on the the work of Viola [76] and initially used integral images. Five approaches were initially investigated, standard deviation, absolute deviation, pyramid reduction with both previous methods, a sobel 2nd order filter and finally approximate absolute deviation. Three methods were chosen to generate the image masks. Method 1 generates Standard deviation using Integral Images with `sum` and `square sum` matrices on 32x32 clusters, the size chosen to constrain the Integral mean computations to 16-bit integers, these are further sub-divided, for performance comparison purposes, into smaller 4x4 blocks. Method 2 generates deviation by utilising the absolute difference between sample and mean, avoiding the use of the Integral images `square sum` matrix and subsequent square roots. Method 3 generates an Approximate Absolute deviation by direct computation of a single value, from each of 4 adjacent sub-groups of 4x4 blocks.

Figure 3.1 shows four images generated by the software demonstrator. The original image, Figure 3.1(a), after conversion to a monochrome image was clustered in smaller 4x4 blocks with thresholded deviation mask applied to the gray scale matrix of the original image (< threshold is black and non-significant, >= threshold is the corresponding pixel grey level and significant). A variable number of clusters per image 8x8, 16x16 can also be applied with Power, Performance, Quality (PPQ) trade-offs.

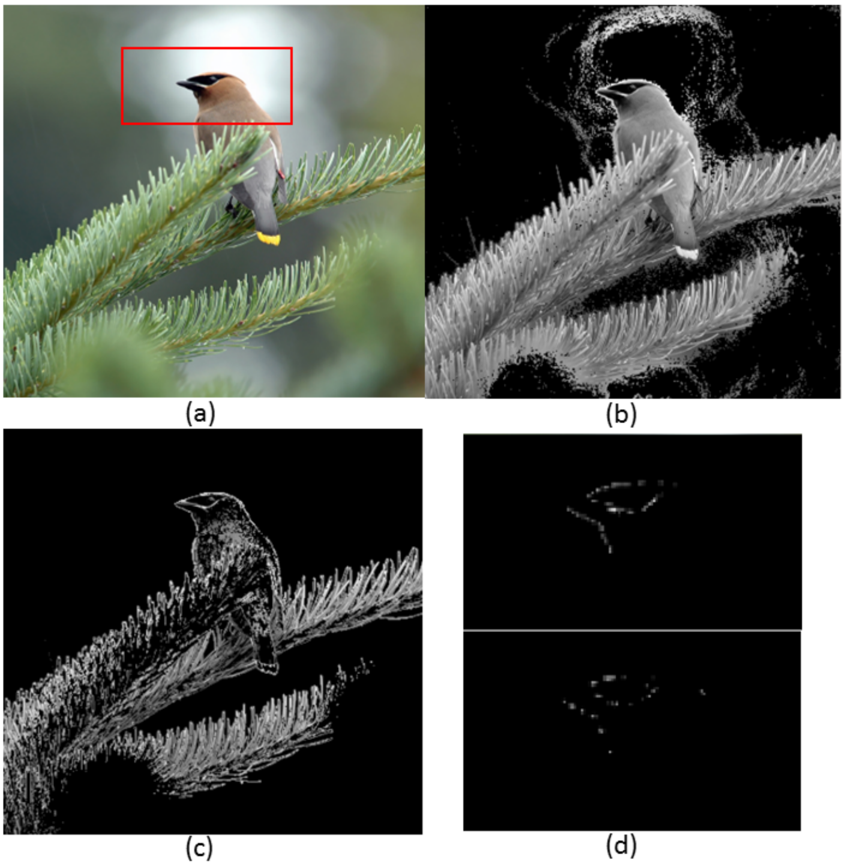


Figure 3.1: Significance of image with different threshold levels. (a) is original image. Red area detail is enlarged in (d). (b) Methods 1,2 & 3, threshold = 3. (c)Methods 1,2 & 3, threshold = 20. (d) Red area Top Methods 1 & 2, Lower 3, threshold =120

Figure 3.1(b) and (c) demonstrate that at low threshold levels, 3 and 20, the deviation figures for each block using the three methods don't show immediately discernible differences in the image masks. Figure 3.1(d) shows a zoomed-in red area of image (a) with a threshold of 120. The top image shows little difference between Methods 1 and 2, the lower image shows the sparser image results of Method 3. Method 1 utilises the compute-intensive OpenCV function *integral()* to generate Integral and Square Sum matrices and subsequent *sqrt* operations, leading to up to 180 ms latency per 20Mpixel image. Method 2 utilises the integral() function to generate the Integral matrix only and then uses the less intensive `abs()` function to generate absolute variance. This reduced the latency to ≈160 ms. Method 3 in Figure 3.1(d) used only one sample from each 4x4 block to compute an approximate absolute deviation using simplified summation, with only ≈ 6ms latency per image.

Varying the thresholds can generate optimistic (too few significant blocks) or pessimistic (too many significant blocks) outcomes. This will be used as a control knob for meeting specified quality requirements in our proposed approach.

The calculation of standard deviation is compute intensive, requiring squares and square roots. As the fundamental point of the research project is to explore ways of reducing processing effort on UHD images (>2kpixel width), the research shifted to ways of investigating a reduction in the compute intensity and power consumption. This led to exploratory work with a C++, OpenCV3.1 based demonstrator in the following areas:-

- Standard deviation based on equation 3.1, computed using Integral Image functionality.

$$\sigma = \sqrt{\frac{\sum_{j=1}^{N}(x_j - \mu)^2}{N-1}} \tag{3.1}$$

- Utilising Absolute Deviation based on equation 3.2, instead of standard deviation, removing the requirement for square and square root computations, reducing computation time and effort.

$$\sigma = \frac{\sum_{j=1}^{N} \mid x_j - \mu \mid}{N-1} \tag{3.2}$$

- Reduction of Original image size by pyramid reduction by a factor of 16, utilizing the OpenCV pyrDown() function.
- Investigation of the Sobel filter second order differential transform.
- Generating an Approximate Absolute Deviation concept based on sub-sampling the image pixels.

Further details on each of how these phases are generated and the timing results will be discussed as follows.

3.5 STANDARD DEVIATION METHOD

Calculating standard deviation with Integral images is a methodology heavily used in facial recognition and other OpenCV functionality. While Viola's paper [76] explains how the Integral Image Sum and Square sum matrices are generated and can be used to generate rectangle sums, there is no further elaboration of how the Integral matrices can be used to generate a mean and deviation calculation. It was necessary to perform a verification exercise in an excel spreadsheet to ascertain that the calculations will be accurate and that explanation follows.

Fig:- 3.2 shows a small example of an Excel based 'dummy' Image for this exercise. The background and four coloured regions were generated using the Excel RANDBETWEEN() function to create areas with differing means and standard deviation. At the bottom left of this figure are shown the excel calculated values of mean and standard deviation for the whole matrix area and each of the four coloured sub areas, the results were generated with the Excel AVERAGE() and STDEV() functions. It needs to be highlighted at this stage that the Excel STDEV() function utilises Sample standard deviation and not population standard deviation i.e. applies Bessel Correction using 'n-1' as a divisor rather than 'n'.

Fig:- 3.3 shows the RANDBETWEEN() values used for the five areas of the dummy image.

The next stage is to generate the Integral Image Sum and Square sum matrices. In this process both matrices are increased in dimension by one row and one column, in row and column zero, with all the values initialised to zero. If I is considered as the original Image Matrix and I' as the newly generated sum. Each pixel element reference I(x,y) in the original image, then corresponds to the calculated sum element I'(x+1,y+1) in the Integral Sum matrix. The individual elements are then calculated starting from I(1,1), x=1, y=1 as

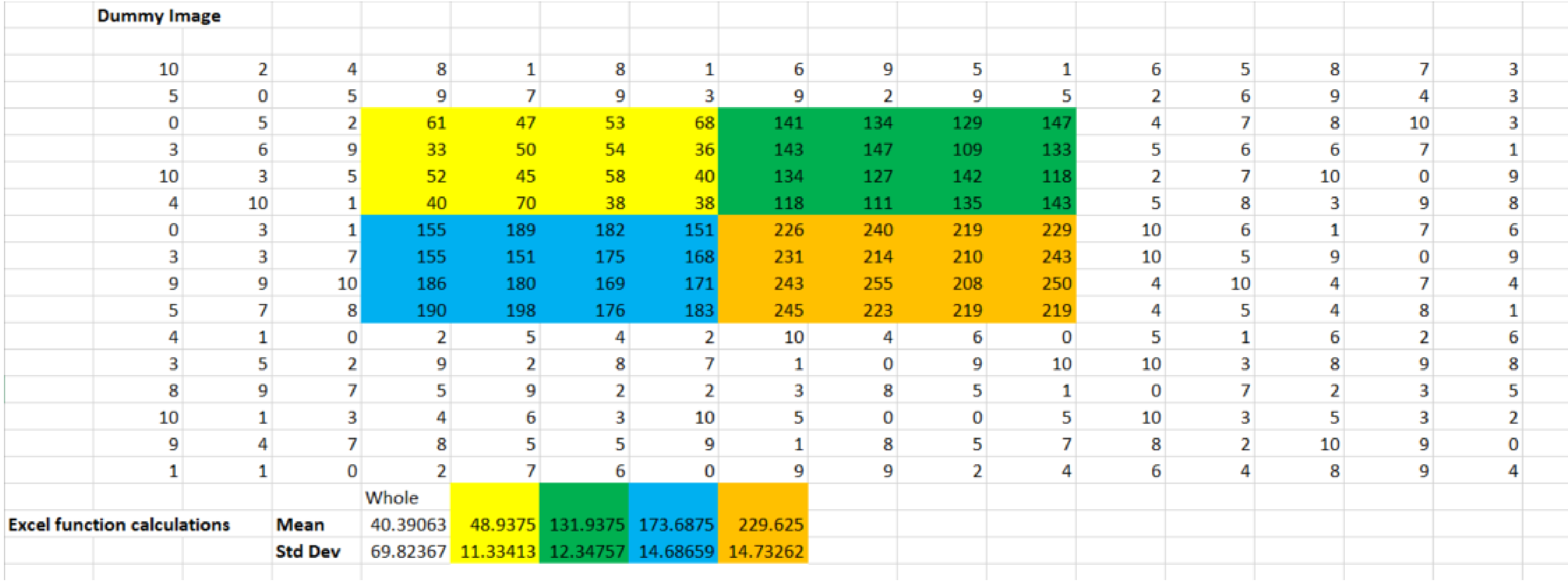
$$I'(x,y) = I(x-1,y-1) + I'(x,y-1) + I'(x-1,y) - I'(x-1,y-1) \tag{3.3}$$

Note that the second and third elements of the above equation also add in the fourth element I'(x-1,y-1) twice, so the one value is subtracted to generate the correct sum.

In order to understand how the generated Image sum matrix is now used to generate means, refer to Fig:3.4.

Each rectangle now has a sum at the bottom right, A-D, representing the sum of everything above and to the left of that cell, so in this case D represents the sum of the whole image. In order to extract the sum for the partial area, coloured light blue, of Area D, bounded by A,B and C, we need the value of the four references A:D and calculate D-C-(B-A), effectively subtracting rectangles. In this calculation the value of area A is contained in each of the sums B and C and is therefore subtracted twice so one value of A is added back in to give the correct total for the sub area D.

Fig: 3.5 shows the Integral Sum Matrix which is now used to generate the mean of each of the five areas, whole area and four coloured areas, utilising the Integral Image Sum Matrix. This matrix shows the incremental Integral sum from top left to bottom right. At the bottom



	Dummy Image															
	10	2	4	8	1	8	1	6	9	5	1	6	5	8	7	3
	5	0	5	9	7	9	3	9	2	9	5	2	6	9	4	3
	0	5	2	61	47	53	68	141	134	129	147	4	7	8	10	3
	3	6	9	33	50	54	36	143	147	109	133	5	6	6	7	1
	10	3	5	52	45	58	40	134	127	142	118	2	7	10	0	9
	4	10	1	40	70	38	38	118	111	135	143	5	8	3	9	8
	0	3	1	155	189	182	151	226	240	219	229	10	6	1	7	6
	3	3	7	155	151	175	168	231	214	210	243	10	5	9	0	9
	9	9	10	186	180	169	171	243	255	208	250	4	10	4	7	4
	5	7	8	190	198	176	183	245	223	219	219	4	5	4	8	1
	4	1	0	2	5	4	2	10	4	6	0	5	1	6	2	6
	3	5	2	9	2	8	7	1	0	9	10	10	3	8	9	8
	8	9	7	5	9	2	2	3	8	5	1	0	7	2	3	5
	10	1	3	4	6	3	10	5	0	0	5	10	3	5	3	2
	9	4	7	8	5	5	9	1	8	5	7	8	2	10	9	0
	1	1	0	2	7	6	0	9	9	2	4	6	4	8	9	4
				Whole												
Excel function calculations			**Mean**	40.39063	48.9375	131.9375	173.6875	229.625								
			Std Dev	69.82367	11.33413	12.34757	14.68659	14.73262								

Figure 3.2: Integral image dummy frame.

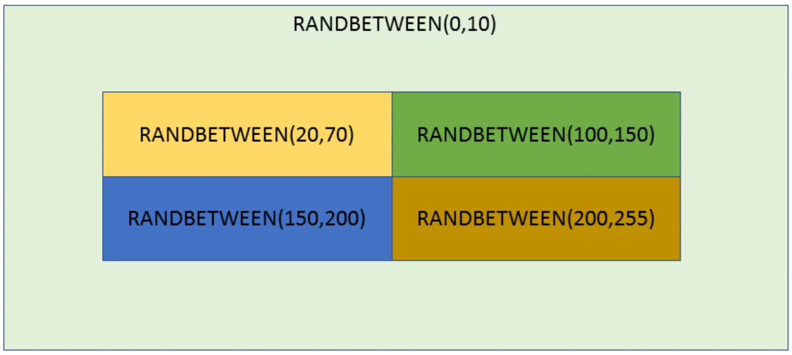


Figure 3.3: Construction of integral image areas.

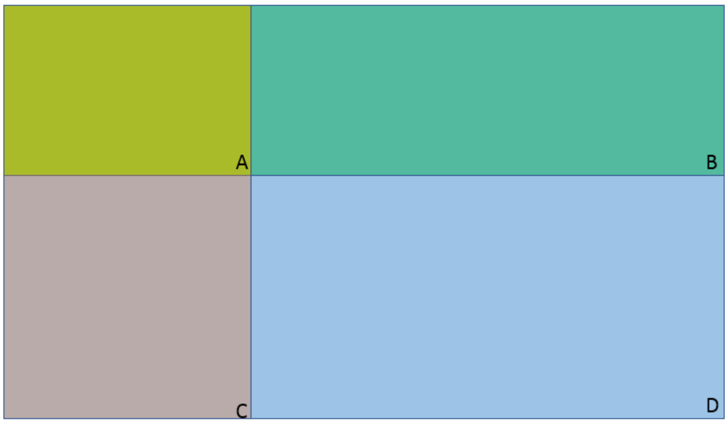
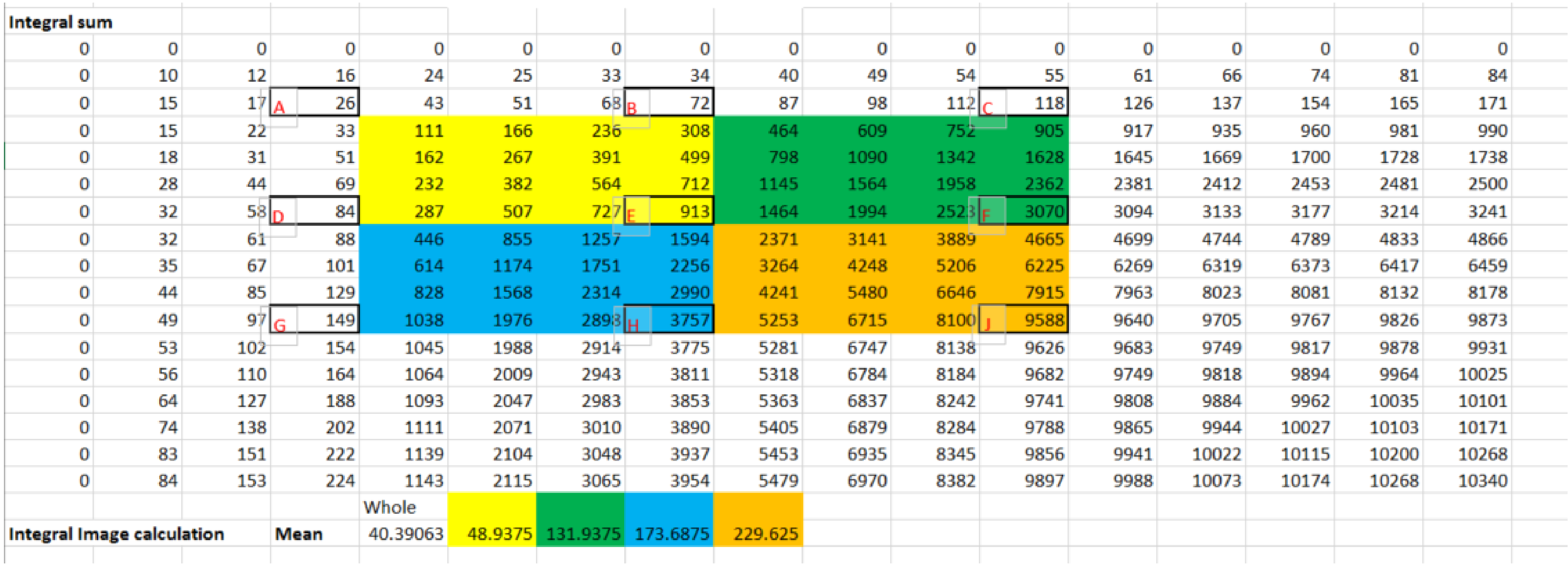


Figure 3.4: Calculation of integral image area sum.

left, below the matrix, are shown the five calculated means for each area section. The whole area total is the value of the bottom right value and the mean calculated through division by the cell count. For the four coloured areas in the centre of the sum matrix, can be seen highlighted Values, labelled A-J. These nine look up values are used to calculate each of the four area sums, demonstrating the relative efficiency of this technique when processing multiple RoIs. How the sum of the RoI is calculated is covered in the Viola paper [76] but is repeated here.
Each value in the integral sum represent the total of the original image pixels above and to the left of the pixel being referenced Table: 3.1 Shows the A-J values and the excel calculations to generate the four individual means for these areas.

Fig: 3.6 shows the Sum of squares matrix, generated in an identical fashion to the sum matrix but adding in the square of the pixel value elements. The nine squares sums A2-J2, which are used, along with the corresponding Sum matrix values, to generate the variance values for each of these areas. The variance is calculated by

$$Var(x) = 1/(n-1)(S2-(S1^2)/n) \tag{3.4}$$



Integral sum																
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	10	12	16	24	25	33	34	40	49	54	55	61	66	74	81	84
0	15	17	A 26	43	51	68	B 72	87	98	112	C 118	126	137	154	165	171
0	15	22	33	111	166	236	308	464	609	752	905	917	935	960	981	990
0	18	31	51	162	267	391	499	798	1090	1342	1628	1645	1669	1700	1728	1738
0	28	44	69	232	382	564	712	1145	1564	1958	2362	2381	2412	2453	2481	2500
0	32	58	D 84	287	507	727	E 913	1464	1994	2523	F 3070	3094	3133	3177	3214	3241
0	32	61	88	446	855	1257	1594	2371	3141	3889	4665	4699	4744	4789	4833	4866
0	35	67	101	614	1174	1751	2256	3264	4248	5206	6225	6269	6319	6373	6417	6459
0	44	85	129	828	1568	2314	2990	4241	5480	6646	7915	7963	8023	8081	8132	8178
0	49	97	G 149	1038	1976	2893	H 3757	5253	6715	8100	J 9588	9640	9705	9767	9826	9873
0	53	102	154	1045	1988	2914	3775	5281	6747	8138	9626	9683	9749	9817	9878	9931
0	56	110	164	1064	2009	2943	3811	5318	6784	8184	9682	9749	9818	9894	9964	10025
0	64	127	188	1093	2047	2983	3853	5363	6837	8242	9741	9808	9884	9962	10035	10101
0	74	138	202	1111	2071	3010	3890	5405	6879	8284	9788	9865	9944	10027	10103	10171
0	83	151	222	1139	2104	3048	3937	5453	6935	8345	9856	9941	10022	10115	10200	10268
0	84	153	224	1143	2115	3065	3954	5479	6970	8382	9897	9988	10073	10174	10268	10340
				Whole												
Integral Image calculation			**Mean**	40.39063	48.9375	131.9375	173.6875	229.625								

Figure 3.5: Integral image sum matrix.

Minimising mean calculations example

A	26	B	72	C	118
D	84	E	913	F	3070
G	149	H	3757	J	9588
Sum1	783		E-D-(B-A)		
Sum2	2111		F-E-(C-B)		
Sum3	2779		H-G-(E-D)		
Sum4	3674		J-H-(F-E)		
Mean1	48.9375	Sum1/COUNT(E25:H28)			
Mean2	131.9375	Sum2/COUNT(I25:L28)			
Mean3	173.6875	Sum3/COUNT(E29:H32)			
Mean4	229.625	Sum4/COUNT(I29:L32)			

Table 3.1: Excel calculation of the four integral image means from the sum matrix.

with Bessel correction applied, where $S2 = sumofsquares$ and $S1 = integralsum$. Table: 3.2 shows the extracted values and the calculations for the four variances. Again the whole image variance is calculated solely from the lower right value along with it's corresponding Integral sum value.

Minimising variation calculations example

A2	170	B2	520	C2	854
D2	576	E2	41171	F2	322312
G2	1053	H2	527561	J2	1655600
SumSq1	40245		E2-D2-(B2-A2)		
SumSq2	280807		F2-E2-(C2-B2)		
SumSq3	485913		H2-G2-(E2-D2)		
SumSq4	846898		J2-H2-(F2-E2)		
	Variance	Formula			
Var1	128.4625	1/(COUNT(E46:H49)-1)*(((SumSq1)-((Sum1_)^2)/COUNT(E25:H28)))			
Var2	152.4625	1/(COUNT(I46:L49)-1)*(((SumSq2)-((Sum2_)^2)/COUNT(I25:L28)))			
Var3	215.6958	1/(COUNT(E50:H53)-1)*(((SumSq3)-((Sum3_)^2)/COUNT(E29:H32)))			
Var4	217.05	1/(COUNT(I50:L53)-1)*(((SumSq4)-((Sum4_)^2)/COUNT(I29:L32)))			
	Sigma	Formula			
SD1	11.33413	SQRT(Var1)			
SD2	12.34757	SQRT(Var2)			
SD3	14.68659	SQRT(Var3)			
SD4	14.73262	SQRT(Var4)			

Table 3.2: Calculation of integral image variance from square sum and sum matrices.

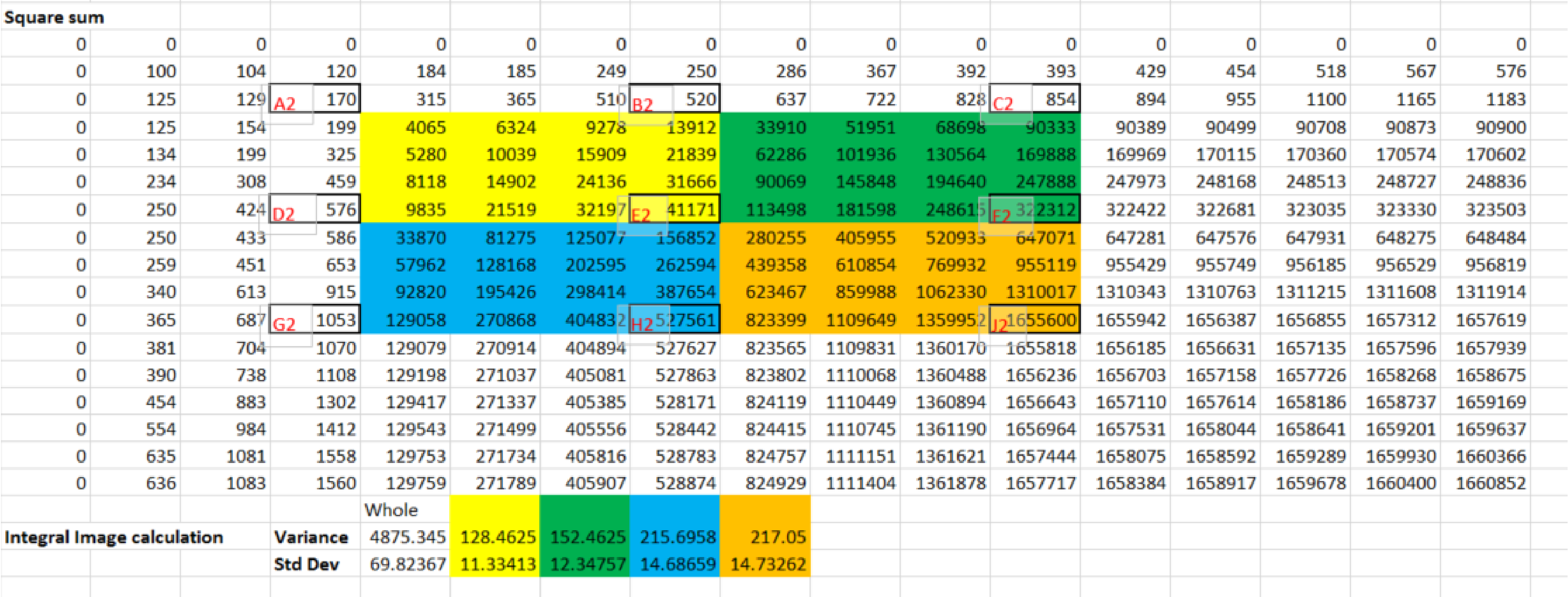
All that remains is to calculate the square root of the Variance to yield the Standard Deviation. It can be seen by comparing the Integral Image calculations of the mean, variance and standard deviation in Fig: 3.6 produce the same results as the excel functional values calculated in Fig: 3.2.

The OpenCV function integral() is utilized to generate 32x32 Integral and Square Sum matrices. The matrix size was chosen to try and constrain the Integral mean computations to 16bit integers for the initial research.

The 32x32 Integral matrices were then used to generate pre-selectable sub array sizes of NxN where N is one of 4,8 or 16. The results of the mean and standard deviation for each sub array were then written to an equivalent image size mask matrix, using a rectangle fill over the sub array size within the appropriate area of the mask array, this was purely to give an easy masking process with the mask the same size as the image. The mask array was then used with a threshold mask slider bar value to generate an image/mask combination to demonstrate and compare the effects of sub array size, mask value and calculation time for the image in real time. Flowchart Fig: 3.7 indicates the steps required in calculation of the standard deviation results in this process.

3.6 absolute deviation method

The traditional standard deviation value utilises square values to avoid the generation of negative numbers in the course of generating the average deviation. After the average variation is found the value has to be square rooted to generate the deviance value. This means that due to the squaring the furthest outliers from the mean have a greater effect on the overall standard deviation. There is considerable discussion and argument as to the use of either standard or absolute deviation, Leys [78]. Absolute deviation can be calculated from the simple function, deviation = abs(value-mean) utilising the C library function abs(value). This removes the necessity for squaring and iterative square root functions and is a much simpler operation for a processor to perform leading to a reduction in the time difference between calculation of standard and absolute deviation. The flow chart for Absolute deviation is structurally similar to the standard deviation, Fig. 3.7, the main differences being that in the first process box the square sums are not required and in the second box Absolute deviation is calculated, thus removing the necessity for squaring and square root calculations thereby reducing computational effort.



Square sum																
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	100	104	120	184	185	249	250	286	367	392	393	429	454	518	567	576
0	125	129	170	315	365	510	520	637	722	828	854	894	955	1100	1165	1183
0	125	154	199	4065	6324	9278	13912	33910	51951	68698	90333	90389	90499	90708	90873	90900
0	134	199	325	5280	10039	15909	21839	62286	101936	130564	169888	169969	170115	170360	170574	170602
0	234	308	459	8118	14902	24136	31666	90069	145848	194640	247888	247973	248168	248513	248727	248836
0	250	424	576	9835	21519	32197	41171	113498	181598	248615	322312	322422	322681	323035	323330	323503
0	250	433	586	33870	81275	125077	156852	280255	405955	520933	647071	647281	647576	647931	648275	648484
0	259	451	653	57962	128168	202595	262594	439358	610854	769932	955119	955429	955749	956185	956529	956819
0	340	613	915	92820	195426	298414	387654	623467	859988	1062330	1310017	1310343	1310763	1311215	1311608	1311914
0	365	687	1053	129058	270868	404832	527561	823399	1109649	1359952	1655600	1655942	1656387	1656855	1657312	1657619
0	381	704	1070	129079	270914	404894	527627	823565	1109831	1360170	1655818	1656185	1656631	1657135	1657596	1657939
0	390	738	1108	129198	271037	405081	527863	823802	1110068	1360488	1656236	1656703	1657158	1657726	1658268	1658675
0	454	883	1302	129417	271337	405385	528171	824119	1110449	1360894	1656643	1657110	1657614	1658186	1658737	1659169
0	554	984	1412	129543	271499	405556	528442	824415	1110745	1361190	1656964	1657531	1658044	1658641	1659201	1659637
0	635	1081	1558	129753	271734	405816	528783	824757	1111151	1361621	1657444	1658075	1658592	1659289	1659930	1660366
0	636	1083	1560	129759	271789	405907	528874	824929	1111404	1361878	1657717	1658384	1658917	1659678	1660400	1660852
				Whole												
Integral Image calculation			Variance	4875.345	128.4625	152.4625	215.6958	217.05								
			Std Dev	69.82367	11.33413	12.34757	14.68659	14.73262								

Figure 3.6: Integral image sum of squares matrix.

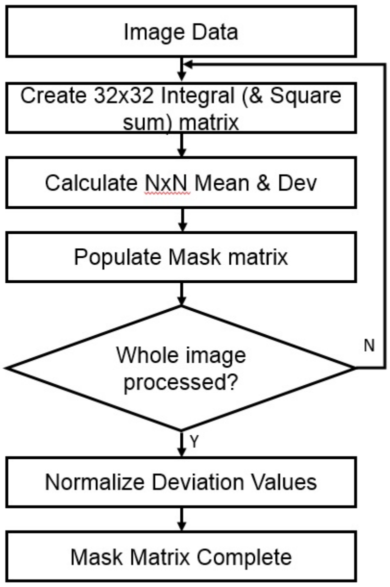


Figure 3.7: Flowchart for absolute (and standard) deviation utilising integral images.

3.7 pyramid reduction method

The OpenCV pyrDown() function reduces an image size by a maximum factor of two in each dimension. In order, therefore, to reduce each image dimension by a factor of 4, the process has to be repeated twice to generate an image pyramid, see Fig. 3.8. However this does add significant time to the initial reduction but a much greater time saving is made on the reduced image deviation computations. This process acted as a good demonstrator that the Approximate Absolute

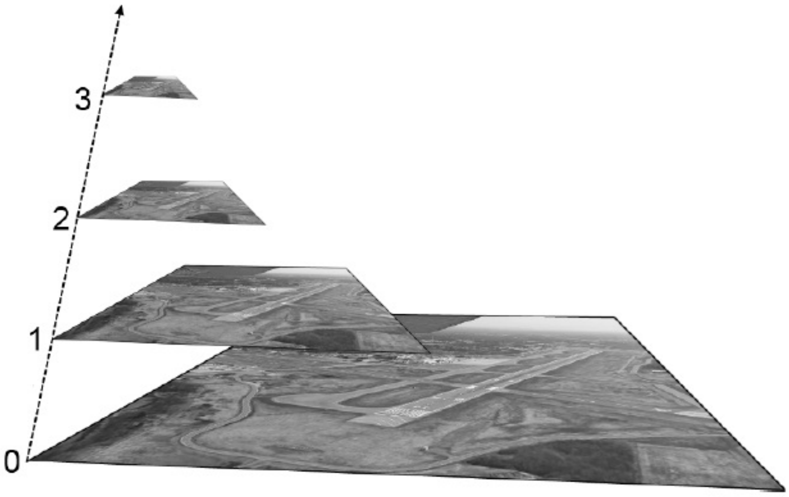


Figure 3.8: Pyramid image concept, the larger image is reduced by a factor of 2 in stages.

Deviation route was worth investigation and would probably be a viable solution.

3.8 sobel 2nd order derivative method

After the initial demonstrator software development, thoughts turned to thinking about other ways of discovering significance. One other technique that came to mind was the second order derivative. It turns out this can easily be achieved in OpenCV by using the second order Sobel filter. However the result is slightly noisier, Fig. 3.9 and initial experimentation showed lack of definition and dynamic threshold range that is available with the standard and absolute deviation masks. The computation time was only slightly quicker than the standard deviation run time, roughly equivalent to the absolute deviation run time. For these reasons any further progression with the Sobel filter had been discounted for this thesis.

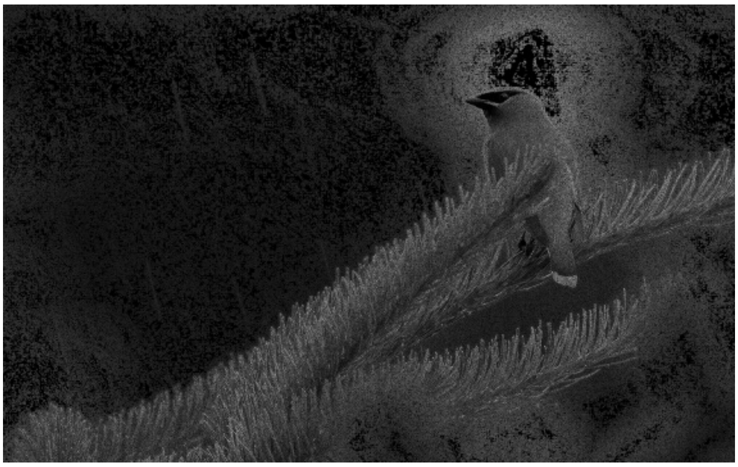


Figure 3.9: Sobel 2nd order derivative with low threshold value.

3.9 approximate absolute deviation method

For the approximate calculation, one pixel is sampled in each of a 4x4 image sub area and used to generate a local mean and absolute deviation between a group of 4 (2x2) sample sub areas,(4 samples out of 64). This is performed as a two pass methodology, the first pass reads the values of two full image rows into a 2 X(width/4) sub matrix, the local mean and absolute deviations are calculated and written to the appropriate two rows of the full size mask matrix. This is then repeated for the remaining pairs of row values extracted from the image, in effect the means and approximate absolute deviation are calculated across the image width for the two sub-sampled image rows in turn. The code was written to take best advantage of main memory accesses, line fills, efficient cache usage and utilization of right shifts instead of division as divisors are straight powers of two. Fig. 3.10 shows the processing steps required.

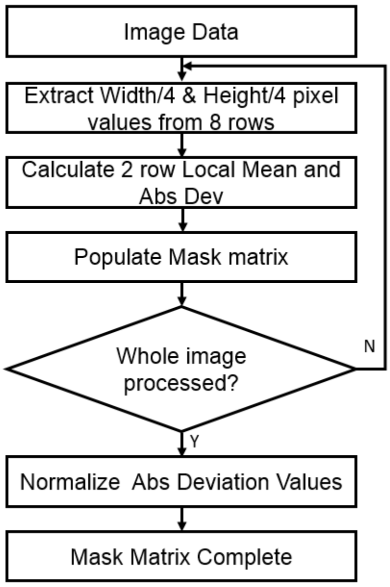


Image Data
Extract Width/4 & Height/4 pixel values from 8 rows
Calculate 2 row Local Mean and Abs Dev
Populate Mask matrix
Whole image processed?
N
Y
Normalize Abs Deviation Values
Mask Matrix Complete

Figure 3.10: Calculation flowchart for approximate absolute deviation.

3.10 experimental results

The comparative timing results were all generated utilising a 4x4 sub-array computation during the various forms of deviation calculation, in order to offer a fair timing comparison between the different processes. In reality for a large UHD image >2k width, 16x16 sub arrays generate adequate results in a shorter period but the approximate absolute deviation gives acceptable results in a much shorter time with a currently in-built 4x4 resolution. All the following timing information tests were conducted on a 5184 x 3888 pixel (20.1 MPixel) image, with the software running on an Ubuntu based Toshiba Satellite Pro laptop in order to render comparative timing results. The timings were based on computation of the results without image display in order to avoid the vagaries of timing variation due to GUI display in the OS.

3.10.1 *Standard Deviation*

Fig. 3.11 shows the timing for Standard deviation on a full size image with 4x4 sub arrays. These standard deviation figures should be taken as the normal reference that we are trying to improve on. Note that the 'create mask array time' is included in Fig. 3.11 for reference only. The demonstrator program generates full size Mask matrices to demonstrate the masking effect on the original size image when

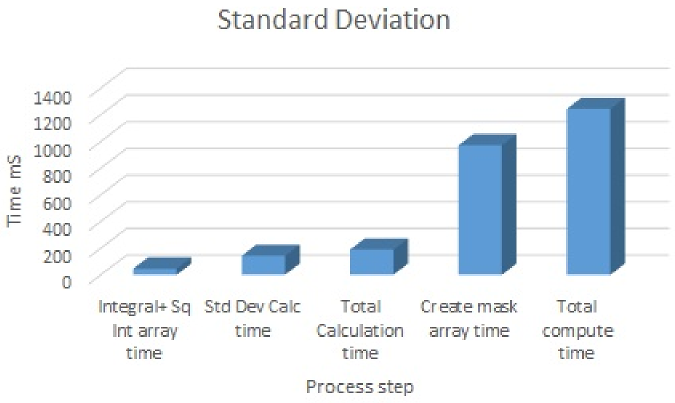


Figure 3.11: Timing for standard deviation run on full image.

raising the minimum threshold, so some of the large time periods seen here would not appear in an end product. Instead a smaller deviation matrix could be used to hold the calculated values for onward significance detection. It was also not possible to isolate some portions of the standard deviation computation elements that were included with the demonstrator masking which would cause slightly longer execution times in an end product.

3.10.2 *Absolute Deviation*

Fig. 3.12 compares the calculation times between Standard and absolute deviation. The three columns, left to right, show the OpenCV

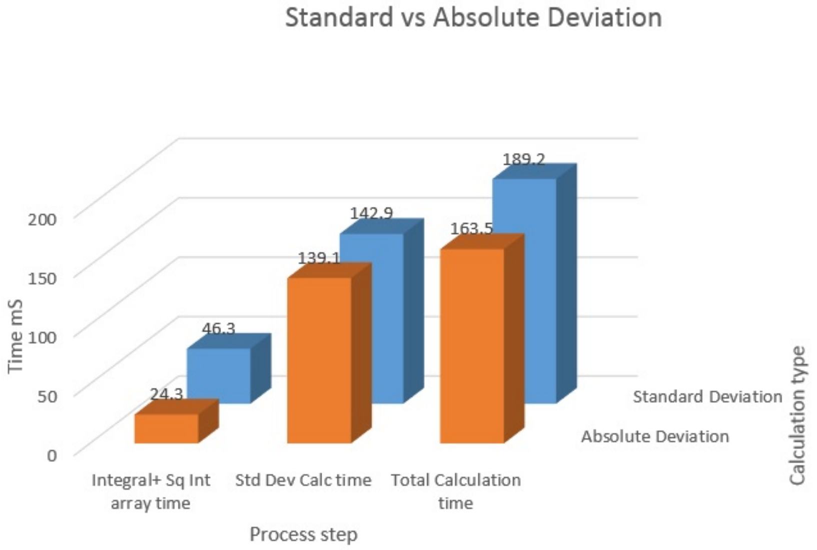


Figure 3.12: Timing for standard vs absolute deviation run on full image.

Integral() execution time, the deviation calculation time and the total time of column 1+2. At this stage no optimisation of the code has been performed as it was merely meant for use as a demonstrator. The first noticeable time difference is that standard deviation requires Integral and Square Sum matrices whereas Absolute deviation only requires the integral matrix to calculate the means. The Calculation time for Absolute deviation is also slightly quicker due to the lack of square and square root computations. The overall time saving on the absolute calculation was in the order of 30mS when compared with the 189.2mS for standard deviation.

3.10.3 *Pyramid Reduced Image Standard and Absolute Deviation*

Fig. 3.13 shows the run times for a reduced image utilising the PyrDown() function to 1/16th of the original image size for both Standard and Absolute deviation. Here there is an extra image reduction time

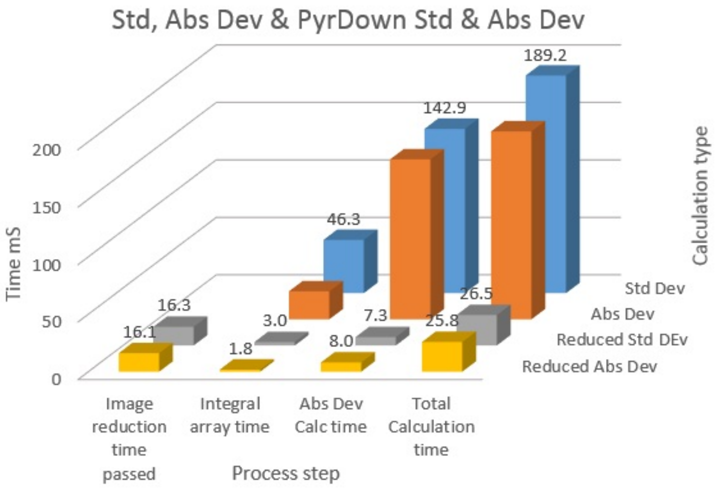


Figure 3.13: Timing, for standard. vs absolute, deviation run on 16x reduced image.

of 16mS appearing due to the extra functionality to perform the PyrDown() function but the overall effect is to significantly reduce both the standard and absolute deviation phase calculation times and the overall calculation time by an order of magnitude, around 26mS for both as opposed to 190mS for Standard Deviation. The image threshold masking showed similar results for both types of deviation results with the Absolute deviation equivalent feature masking occurring at slightly lower values.

3.10.4 *Approximate Absolute Deviation*

Fig. 3.14 compares the approximated absolute deviation against the previous full image standard and absolute deviation.

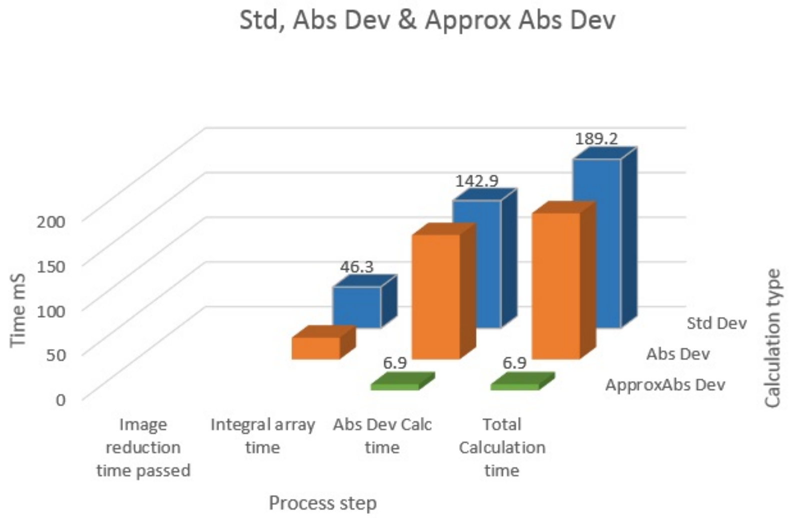


Figure 3.14: Timing for approximate vs absolute & standard, deviation run.

The much reduced calculation time for the approximated Absolute Deviation is now around two orders of magnitude quicker than the original standard and absolute deviation for a 20MPixel image. Overall, comparing the run time of the original whole image standard deviation of 189mS with the run time of the approximate Absolute deviation of 6.8mS represents a considerable computational and energy saving.

In order to give an appreciation of why we see such time differences between the standard, absolute and approximate absolute calculations the following explanation will reinforce the reasoning. In a generic computer system, operations on data constitute two elements:

1. Data movement, ie fetching data from main memory, a read operation, to the cache for the processor to access the data locally or alternatively transferring the data in the cache back to main memory, a write operation. This process is typically accomplished as a batch of 8 64 bit words, known as a line fill, in order to optimise DDR-DRAM memory usage.
2. Data processing, where the data in cache is transferred to the processor, the required functions performed and the resultant data is written back to the cache, to await transfer back to main memory.

In order to give an understanding of the computation performed during the three main deviation calculations and highlight the relative

	Main Memory Data Movements		Maths Operations				Compute Time ms	
Deviation calculation	read line fills	write line fills	add/ sub	multiply	divide	sqrt	Integral Image	Total
standard	64	64	29699	12288	8449	4096	46.3	189.2
absolute	64	64	16387	0	1	0	24.3	163.4
Approximate Absolute	16	4	448	0	64	0	NA	6.9

Table 3.3: Table of operations.

computational demands of the standard, absolute and approximate absolute deviation, Table 3.3 breaks the operations down into a simplified number of data movements and maths processes that are performed on a single 64x64 pixel equivalent work-group area to generate the significance values by the three deviation calculations.

In the data movement columns we see that in the exact calculations there would be 64 line reads to extract the pixel information of the particular segment of the image, bearing in mind that one line fill will transfer 64 bytes of information, so 64 line fills will render 64x64 bytes. In the case of the approximate calculation, assuming we are using a 4x4 sub-group, we only need to transfer in every fourth row values and the subsequent calculation extracts every fourth value to generate a local mean as previously explained in Section 3.9. Hence we see an equivalent figure of 16 line fill reads and only 4 line fill writes.
Operations used in the calculation of the three types of local deviation values are as follows:

- Standard Deviation utilises the Integral sum and Integral square matrices, the calculations are quite involved and are as follows. For each element of the sum and square matrices there are two adds and one subtract during generation. 64 x 64 x 3 =12288 add/subtract operations each. The square value generation involves a squaring of each value before the add/subtract, ie. 4096 multiplies. Generation of each of the local means involves 3 add/subtract and 1 divide for 256 values ie 768 add/subtract and 256 divides. Generation of variance, first generate the square sum for each 4x4 sub-array from the Integral square matrix, 3 Add/subtract operations for 16 x 16 sub-arrays, 16 x 16 x 3 =768. Next generate the 4096 variance values, which for each value involves, 1 square ie. multiply, 4096 operations, 1 subtract 4096, 2 multiplies, 8192 2 divisions 8192. Finally generate standard deviation for each value by square rooting the variance, 4096. Giving the following totals:
  Add/subtract 12288 + 12288 + 3 + 256 + 4096 + 768 = 29699.
  Multiply; 4096 + 8192 = 12288.

Divide; 8192 + 256 + 1 = 8449.
Square root; 4096.

- Absolute deviation is a relatively simpler operation and utilises only the Integral sum matrix. Generating the sum matrix requires 3 add/subtract operations for each value in the 64x64 array 1̄2288. It then generates a local mean for the whole 64x64 matrix 3̄ add/subtract operations with 1 division. It then generates an absolute deviation by simply differencing every value in the array from the mean = 4096 add/subtract operations. ie 12288+3+4096 = 16387 add/subtract and 1 divide.

- Approximate Absolute deviation reduces the 64x64 array down to a 8x8 array of approximate values which then utilises 3 additions and 1 division to generate each mean of 4 values in an 8x8 work item. There are then 4 differences generated for the deviation values, giving a total of 7 add/subtracts and 1 divide for each of the 8x8 values.

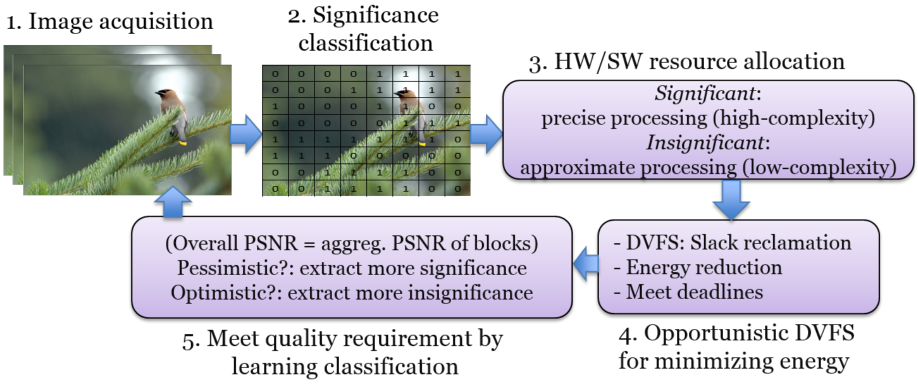
Once the calculation is finished, the calculated values are then transferred out into the main memory array that holds the calculated significance values. This process is repeated for every work-area in the image which for the example Cedar waxwing image we use here, consists of 1000 work-areas. To the right of the table we show the timings from Fig. 3.14 that indicate the relative calculation times of the three methods. Note that the software was an early demonstrator, not optimised for speed, that creates an image mask the same size as the original image, which consumes a considerable amount of computational effort. A more efficient system was created for the demonstrator software discussed in Chapter: 5. The difference in the Integral Image processing time is shown for Standard deviation, using Sum and Square sums, as opposed to the time to generate a Integral sum for Absolute deviation. The Integral sums are not utilised by the Approximate method. The Total deviation calculation time illustrates the considerable saving by the Approximation methodology.
It can be seen that absolute deviation provides a lower workload on the system than standard deviation and approximate absolute deviation provides a huge saving on workload than either of the other two deviation calculations.
The approximate absolute deviation is based on sampling one value in every 4x4 sub area, effectively reducing the Cedar waxwing example from 2560 x 1600 down to a more efficient 640 x 400 area for significance calculations. This scenario instigated the idea of increasing the sub area size for larger >4k images to 8 x 8 or 16 x 16 which will be addressed in Chapter: 5 .

3.11 case study

An opportunity was taken to perform a separate case study, as part of an undergraduate project, in parallel to this research to evaluate the effectiveness of this approach and moving it forward to hardware GPU implementation. This case study built on the approaches of Totoni [79] & Bui [80] and demonstrated a software variable-kernel based parallel convolution filter using machine learning techniques to increase power efficiency and demonstrate application of DVFS, running on an Odroid XU-4 platform consisting of 8 CPU cores, A15 and A7 in a big.LITTLE configuration alongside a MALI T628 MP6 GPU core. This approach has proven useful in generating performance data and further system development application concepts.

3.12 transition to adaptive approximate computing

From the above experiments, and experience gained from the case study, it was proposed to explore the use of optimised Machine Learning algorithms, Venkataramani [30] & [29], and Raha[43]. Runtime management and energy efficiency methodology will be explored to take advantage of the lower computation requirements of the approximation methods. This will utilise Approximate Absolute Deviation with three or four adaptable significance levels. The controlling inputs will be a defined Energy usage target and an expected output image quality, ie two control knobs, Fig. 3.15. Primarily energy usage predictions can be used to set significance threshold levels to identify the areas of an image that hold most significance. This could allow discrimination between areas that are either background or foreground, thereby identifying areas that qualify for a range of significance values with higher significance, requiring higher accuracy or full resolution image processing and lower significance areas that should be subject to either a lower accuracy processing or less frequent update or a combination of both. Secondly the Quality of the image output will act


1. Image acquisition
2. Significance classification
3. HW/SW resource allocation
Significant:
precise processing (high-complexity)
Insignificant:
approximate processing (low-complexity)
- DVFS: Slack reclamation
- Energy reduction
- Meet deadlines
4. Opportunistic DVFS for minimizing energy
(Overall PSNR = aggreg. PSNR of blocks)
Pessimistic?: extract more significance
Optimistic?: extract more insignificance
5. Meet quality requirement by learning classification

0	0	0	0	1	1	1	1
0	0	0	1	1	1	1	1
1	0	0	0	1	1	0	0
0	0	0	0	0	1	1	1
1	1	1	1	1	1	1	0
1	1	1	0	0	0	0	0
0	0	1	1	1	1	0	0
0	0	1	1	1	1	0	0

Figure 3.15: Proposed adaptive approximate computing approach.

as a feedback control to the Machine Learning input in order to adapt to the best balance of energy usage against processed image quality. Currently PSNR is used to derive the QoS figures for processed image quality, however a more lightweight solution such as Grigorian [54] may be explored or possibly particularly UIQI by Wang[38]. It is anticipated that experimentation and development will initially take place based on a GPU assisted processor system. The project may then evolve to a Significance driven parallel computation approach based on an SoC FPGA platform to enable hardware acceleration of the Adaptive approximation methodology.

3.13 FACTORS AFFECTING SIGNIFICANCE

Significance threshold and Energy usage can be seen to be obvious candidates for use as control knobs in conjunction with a PSNR feed back figure to close the control loop. Intuitively, as larger images (>2K pixel width) are processed, a larger approximation array 8x8 or 16x16 instead of the existing 4x4 are feasible. In the case where GPUs are involved with the image processing, the WorkPackage size may also benefit from adjustment, dependent on the GPU architecture.

3.13.1 *Significance Thresholds*

The case study utilised a simple single threshold scenario where any significant areas above a feedback derived threshold were adaptively processed by a 3x3 kernel filter and those below threshold with a 1x1 filter (effectively no processing), or alternatively above threshold with a more accurate 5x5 kernel and below with a 3x3. This then begs the question about the use of three (2 down to 0), or four thresholds, (3 down to 0) where 0 is least significant. The lower threshold being processed with 1x1 and incremental levels with 3x3, 5x5 and 7x7 kernel filters. This contrasts with the current situation in image processing where a single kernel filter, 3x3, 5x5 or more rarely 7x7 is applied as a sliding window across and down the image matrix where the product of the kernel filter against the selected and surrounding pixels is calculated. Table. 3.4 shows the total multiplies and additions (MAC operations) that are required per each pixel processing. The total for each pixel is dependent on the kernel size where the number of elements, N, is the product of the width and height of the kernel. The kernel generates a sum of weighted values with the number of operations required, calculated as N multiplies followed by N-1 accumulates or additions. Note that the 1x1 operation is considered here as a unity gain multiplication. These figures can be utilised as a basis to generate energy usage cost estimates in a Machine Learning approach. It may be argued that FPUs and GPUs can perform combined multiply and accumulates efficiently in a single processor instruction but this scenario still utilises significant hardware to perform these individual tasks.

Kernel size	multiplies per kernel	accumulates per kernel	Total
1x1	1	0	1
3x3	9	8	17
5x5	25	24	49
7x7	49	48	97

Table 3.4: MAC operations per pixel per kernel size.

By using an approximation method for absolute deviation, we extract, in our examples, one pixel from every 4x4 sub area. and compute a mean for each group of four sub areas in an 8x8 array of sub areas, ie we are generating a mean from four values in a 32x32 pixel image section. The absolute deviation from this local mean for each sub area is calculated. After suitable thresholding the whole 8x8 area can then be allocated for a suitable level of Kernel filtering from 1x1, 3x3, 5x5 etc. For GPU applications the workpackage area is subject to minimum size, dependent on the particular GPU architecture used, in order to achieve maximum efficiency. In an FPGA application this may be able to process individual sub areas but is currently beyond the scope of this thesis.

3.13.2 *Work Area Variation and Approximation Stride*

In GPU operation with OpenCL, the image, Workarea, is divided into workgroups, which is the block of data submitted to the GPU for processing and can differ between GPU architectures. The workgroup is further subdivided in to smaller work-items, which may require to share information between them.

In Convolutional Neural Network (CNN) operation a similar technique to that explained previously, called stride, is used to reduce the number of computations during learning and recognition phase. A stride of 2 will apply a kernel to every other pixel in both column and row. So the approximation stride in this case is effectively four, starting at element (1,1), in order to keep away from image edges, and progressing in order (1,5).... (5,1), (5,5).... and so on over the entire image. As higher dimensional images develop eg 8k image, further research may be needed to investigate increasing the stride size to eight and the effects on image processing.

3.13.3 *Energy Usage & DVFS*

After applying the pixel totals of Table: 2.1 to the per pixel MACs of Table. 3.4 it can be seen that a 5x5 filter applied to a 640x480 image increase from 15 x 10e6 MACs to a total of 514 x 10e6 for a 4320 x 2432 image, which represents a 34 fold increase in computational effort.

Significant reduction of computation and utilisation of, either or both, CPUs, and associated FPUs, and GPUs presents the opportunity to utilise DVFS. The case study illustrated a scenario in which DVFS could be implemented in the case of the GPU. As the time to calculate 3x3 kernel on a workgroup was significantly shorter than that for a 5x5. This generated slack time that allowed queueing of extra 3x3 kernels while the 5x5 was still being processed. Tuning of the thresholds in the approximated deviation was able to optimise calculation to maximise fill up of the workgroup queues. The use of Neon FPU and GPU offer significant speed up of parallel calculations which offered the opportunity to decrease the CPU and GPU operating frequency. Variation of the CPU/GPU voltages separately didn't seem to be an option during the case study as there appears to be a withholding of IP for the Odriod XU4 cpu-set by the manufacturer.

3.14 summary and discussion

By further investigation of the above techniques reinforced by the results of the case study we have a model and the tools for implementation of Significance Driven Adaptive Approximate Computing, the details, along wth the further development, of which will be explained in Chapter: 5. A limitation of the case study was the inability to examine more accurately the runtime and energy usage during the runs to allow comparison of energy usage during CPU and GPU runs. This is addressed in the next Chapter: 4 by utilising OpenCV and Arm Compute Library to render energy and performance results for running convolutions on the CPU, GPU or Neon DSP.

4 ENERGY AND PERFORMANCE CHARACTERISATION IN HETEROGENEOUS SYSTEMS

In this Chapter we investigate energy and performance characteristics to allow comparison and contrast between implementations of image processing convolutions performed primarily on a CPU utilising OpenCV and those performed by executing them in a Neon DSP or Mali GPU, utilising OpenCL via the Arm Compute Library. In Section 4.1 we introduce the odroid XU4 platform which was to be the base platform upon which subsequent work would be based. In Section 4.2 we discuss the rationale for this work and in Section 4.3 we discuss the methodology to enable power measurement of the XU4 platform during execution of software models which until this phase had shown to be a problematic experience. Section 4.4 outlines the strategy for measuring both the run time and energy measurement over a suitable measurement period. This section also demonstrates some exploratory work and some observational guidelines for the actual measurement. The results of the measurements and runs are reported in Section 4.5 along with a number of extra observations to be taken into account. Section 4.6 summarises this work.

4.1 INTRODUCTION

The previously mentioned case studies proved that the Odroid XU4, Fig: 4.1 was a very able Linux platform for experimentation with a variety of homogeneous elements, Multiple CPU, Neon FPU and GPU, but presented a challenge in trying to measure the runtime power whilst executing convolution kernels on these on-board facilities.

Specifications for this platform, briefly include:

- Processor Samsung Exynos5422 Quad ARM A15 2.0GHz Quad A7 1.4GHz.
- 3D Accelerator Mali T628 MP6 OpenGL/OpenCL
- Memory 2Gbyte RAM.
- USB3.0 Host, 2x USB2.0 port
- eMMC module socket, MicroSD Card Slot Flash Storage (up to 64GByte each).
- Fast Ethernet LAN 10/100/1000Mbps.

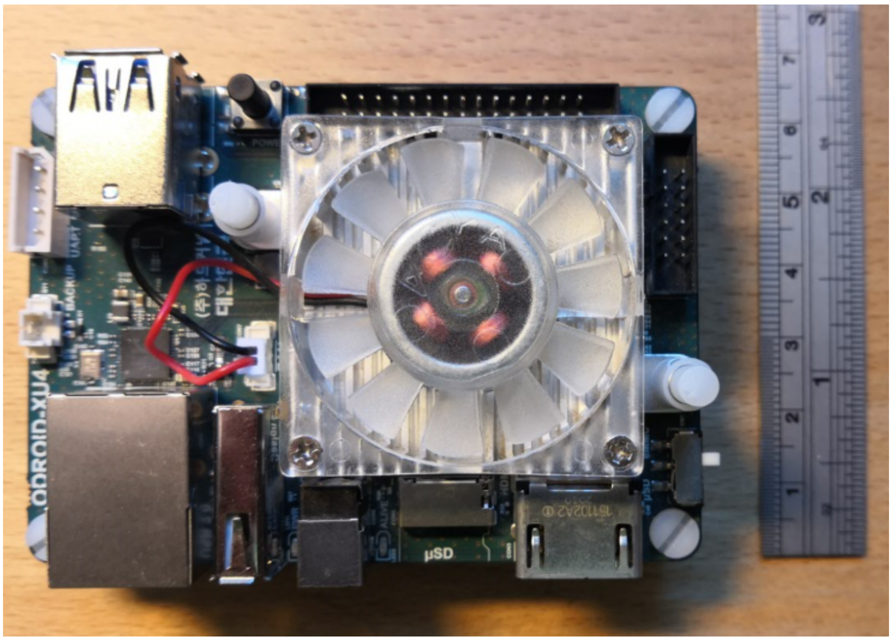


Figure 4.1: The Odroid XU4 platform with Samsung exynos octacore processor and T628 GPU.

- OS Ubuntu 16.04.
- Size 82 x 58 x 22 mm approx.

GPUs, Neon DSPs and sometimes FPGAs are often perceived as power hungry units as opposed to CPUs. While these devices are observed to use a larger quantity of power during the execution phase, the task is finished much quicker so the total energy used in the operation may well be smaller than that during equivalent execution solely on a CPU. It was decided to run 3x3, 5x5 convolutions via ARM Compute Library, in order to enable running the convolutions on both GPU and Neon DSP. In addition 1x1, 3x3 and 5x5 convolutions, using OpenCV were run on the processor only, while simultaneously measuring the Power used by the Odroid XU4 board in all cases.

The first challenge was that, the predecessor to the XU4, the XU3 was the subject of investigation into Power-aware performance adaptation by Aalsaud [81] and had current measurement facilities built in to the board but were not implemented on the XU4. The XU3 utilises Texas Instruments INA231, Inter Integrated Circuit (I2C) communication based current and power monitor, to measure four out of the ten separate board regulated supplies at various voltages. As there is only one 5 Volt supply voltage to the XU4, a Texas Instruments Evaluation module, the INA231EVM, was identified, and delegated to measure the Power input to the board which gives an overall power figure, rather than the XU3's subset of the total power and perhaps misleading picture of the whole power supply scenario.

If we can quantify the amount of power and execution time for a 3x3 or 5x5 convolution on a single pixel, computed on each device, CPU, GPU and Neon DSP, an estimate of the frame processing time and energy consumption for a range of image sizes can be evaluated and help define an evaluation strategy for parallel computation of the various NxN convolutions on a mixture of devices eg 5x5 on GPU, 3x3 on Neon(DSP) and 1x1 on CPU.

4.2 RATIONALE AND PREVIOUS RESEARCH

Among the various challenges in designing a computer system, especially with battery powered heterogeneous embedded targets is the question of how many computing elements, CPU, GPU, FPGA etc. are needed, can we achieve sufficient run time to viably achieve execution of the required task without running out of power reserve?

While most embedded applications have historically utilised low power Cortex-M family there is great interest in adapting vision processing on embedded systems and currently there is evidence of vision processing facilities running on Cortex-M, by OpenMV, Abdelkader [46], albeit they run under MicroPython, while this option may be optimal for demonstrating the concept of embedded vision processing, it is hard to imagine adoption of this language by embedded C/C++ purists in their search for ultimate low power applications. Moves to using Cortex-A systems in order to provide fall detection systems, Nguyen [47], security cameras with night vision, by Abaya [82] and Image Processing Units on low cost embedded hardware, Nair [83], provide evidence of the demand for embedded image processing.

Arm Compute Library released in 2017, provides a number of OpenCV style, along with Machine Learning, functions to run on Cortex-A processors and GPUs, aimed at embedded applications. The Documentation for the ARM Compute Library emphasises the process performance gains by utilising GPU or Neon but there are no clues if the performance could be tuned to reduce energy consumption or what the power demand will be. For this reason it was decided to characterise the performance and power demand of the Compute Library and OpenCV library running on an Odroid XU4 platform, in order to investigate what heterogeneous elements, running under direction of Approximate Significance, could provide the best power-performance trade-offs. As Compute Library is a more recent development, lacking a reasonable user manual, there are not many published papers in this respect.

4.3 POWER MEASUREMENT METHODOLOGY

The Texas Instruments INA231EVM Evaluation Module is a USB based device Fig: 4.2, that comes with a Windows Graphical User Interface (GUI) based software module which enables real time viewing and logging of power usage against time. In this experimental setup the module is configured to utilise the Texas Instruments INA231 device to measure high side power consumption of the Odroid XU4 during software testing.

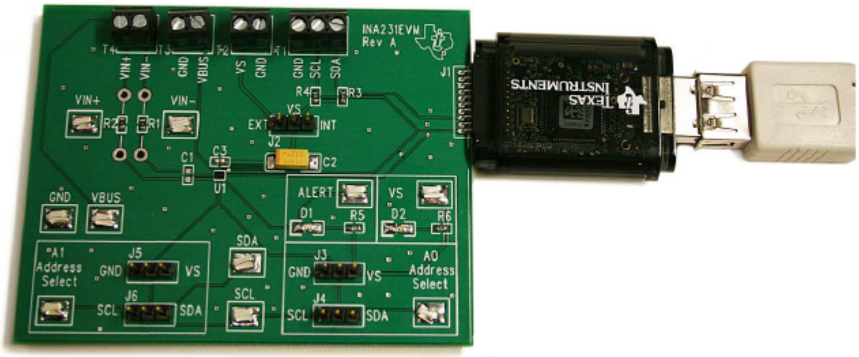


Figure 4.2: The Texas Instruments INA231 evaluation module with the USB interface on the right and the measurement connections shown along the top.

The evaluation module is connected as a high side measurement device, Fig: 4.3, with a 10 milli-Ohm wire wound precision, 1%, current sensing resistor in series with the load. The INA231 device takes two measurements Applied voltage and Shunt Voltage it calculates shunt current and load power from these values. The INA231 also has a programmable conversion time and averaging mode so that multiple measurements can be made and averaged during the USB-I2C measurement periods, typically one second, with the supplied software.

The INA231EVM kit is supplied with GUI based software to drive the device from a Windows based platform, which allows set-up of the device, Fig: 4.4.

The EVM was setup to provide 256 averaged samples per interval in the "Step 2: Configuration Operation" setup box, Averaging Mode was typically set to 256 samples during the one second polling interval. The current shunt value is entered as 10m (milliohms) in the "Step5 Configuration Register" Resistance box and Max Current is set to 6A (Amps). Clicking outside this default value box will see the software generate two values for Low LSB and High LSB. The Low LSB value (183.1uA, microAmps) is entered into the Current LSB box. Near the top of the setup box, "Write All Reg" is clicked, followed by "Read

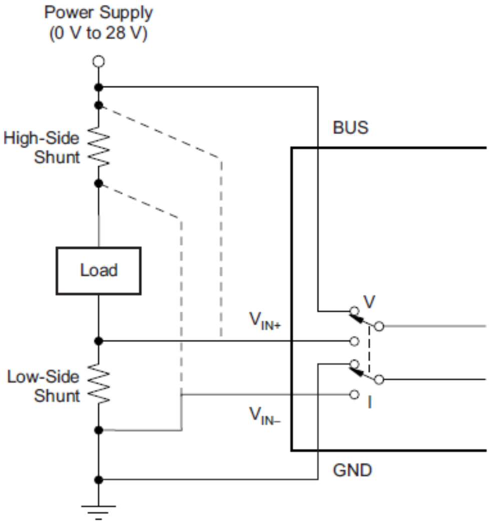
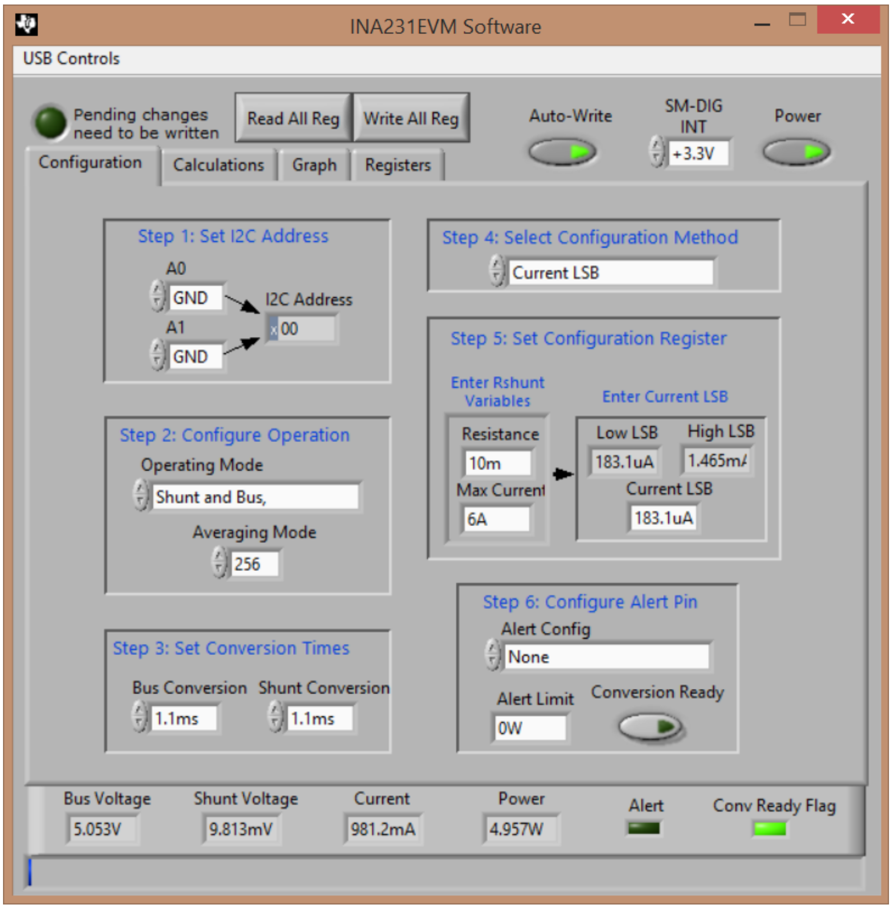


Figure 4.3: Schematic of INA231 load connection schemes.



SM-DIG INT: +3.3V

Step 1: Set I2C Address

A0: GND

A1: GND

I2C Address: x00

Step 2: Configure Operation

Operating Mode: Shunt and Bus,

Averaging Mode: 256

Step 3: Set Conversion Times

Bus Conversion: 1.1ms

Shunt Conversion: 1.1ms

Step 4: Select Configuration Method

Current LSB

Step 5: Set Configuration Register

Enter Rshunt Variables

Resistance: 10m

Max Current: 6A

Enter Current LSB

Low LSB: 183.1uA

High LSB: 1.465m/

Current LSB: 183.1uA

Step 6: Configure Alert Pin

Alert Config: None

Alert Limit: 0W

Conversion Ready

Bus Voltage: 5.053V

Shunt Voltage: 9.813mV

Current: 981.2mA

Power: 4.957W

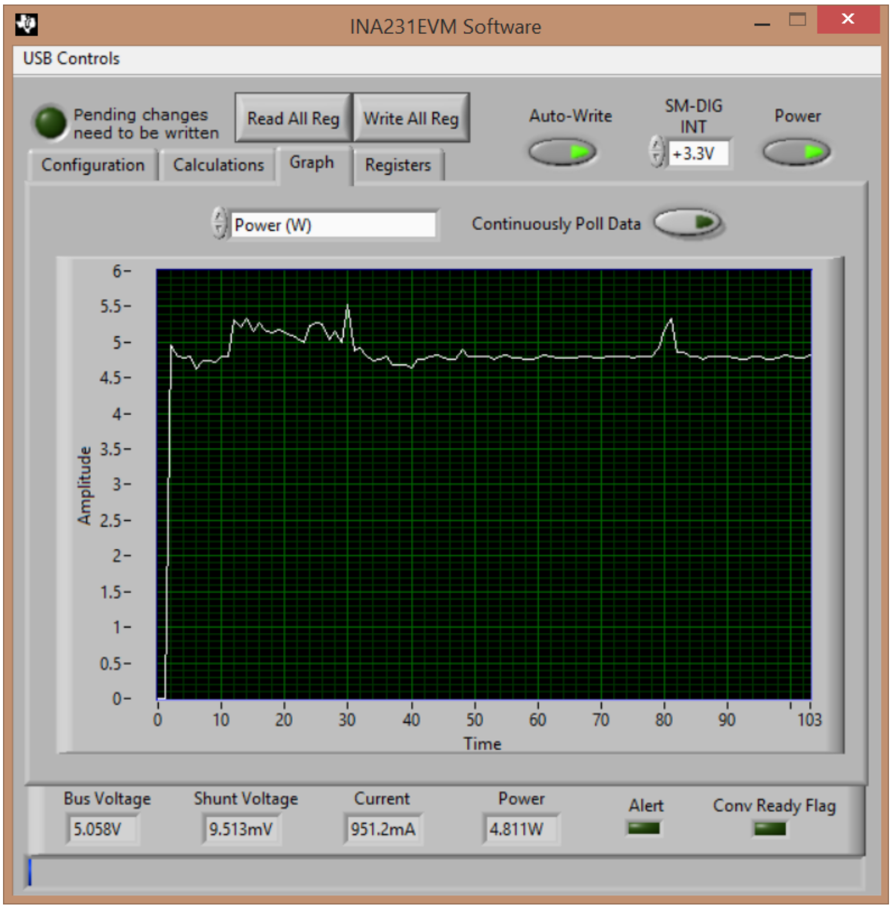
Alert

Conv Ready Flag

Figure 4.4: INA231EVM GUI measurement parameter setup page.

All Reg". The bottom row of boxes at the bottom area of the page should now show typical values of Bus and Shunt Voltage, Current and Power to demonstrate the module is now set up. Selecting the calculations page tab will now show the steps in the calculation of the device energy parameters instead of the default zero values.

Clicking on the Graph tab shows a selectable graph parameter in the drop down box where "Power" is selected. Selecting "Continuously Poll Data" now provides real-time graphing of the Power consumption of the Device Under Test (DUT) and can be stopped and restarted on subsequent "Continuously Poll Data" selections. The EVM software provides graphical display, Fig: 4.5, and export of results in a compatible Comma Separated Variables (CSV) file. This provides a route to export the results into excel and as a quick sanity check, graphing the results in Excel to ensure the resultant graph of the imported data has the same appearance as in the EVM software graph page. This provision allows further viewing and calculation of the performance and energy results.



SM-DIG INT: +3.3V

Power (W)

Bus Voltage: 5.058V

Shunt Voltage: 9.513mV

Current: 951.2mA

Power: 4.811W

Figure 4.5: INA231EVM graphical result viewing page.

4.4 convolution run-time and energy measurement strategy

The power tests were to be executed on the XU4 board which is running Linux and needs to be borne in mind with all power measurements. As the operating system is continuously running and the requirement is to measure the power of the convolution process itself, the image is loaded and the computation only is repeated the required number of times, see Table: 4.1. XU4 linux idle power is around 4.3 Watts with occasional periodic peaks above this, thought to be some Linux system management activity. Occasionally the fan will start to run which consumes around a further 0.8 Watts. In order to retrieve an energy figure for the computation being run we need to measure the total power, subtract the idle power then average the power over the reported run time to give the compute power. The total loop time is then divided by the number of iterations in order to yield the frame time. This figure can then be multiplied by the Compute Power, in Watts, to give the energy per frame, in Joules. This figure can then be further divided by the number of pixels in the frame in order to give an execution time and energy usage for a pixel-kernel, which represents the activity, multiplies and accumulates, required to perform an NxN convolution on a single pixel and it's computational neighbours.

In order to perform this measurement, two special versions of Software were generated to run on Odroid XU4 but could be easily adaptable to other targets:

The first version was for an OpenCV version of software to repeat a number of tight loops, I, of 1x1 convolution (for reference), J, of 3x3 convolution or, K, of 5x5 convolutions across the whole image. I, J and K chosen to give a number of seconds for total loop time. The software outputs to the console which convolution is being run, what size image is being processed, total execution time in milli-seconds and the number of iterations for each convolution. This allows easier alignment of the computation being run with the Power measurements of the INA231EVM spreadsheet output.

The second version was an Arm Compute Library run time module of the above software functionality. As it was not possible to identify an equivalent 1x1 kernel in the Arm Compute Library v17.12 the 1x1 test was excluded, but running, J, 3x3 or K, 5x5 loops on either Neon DSP coprocessor or Mali T628 GPU. Console output information is identical to option 1 with the inclusion of which device, Neon or GPU is being run.

In order to evaluate performance across a range of image dimensions, the target images, of varying sizes were created from a single large resolution, 5184 x 3888, 20Mpixel, still image, with a range of nine sizes from 320 x 240 through to 8192 x 4608 pixels. The 320 x 240 was only run on the GPU and Neon, mainly to explore and emphasise the recommendations by ARM to use GPUs etc on HD images and above only.

Image size		GPU		Neon		OpenCV		
Width	Height	3x3	5x5	3x3	5x5	1x1	3x3	5x5
320	240	40,000	20,000	40,000	20,000	N/A	N/A	N/A
640	480	10,000	5,000	5,000	2,500	2,000	1,000	500
1024	768	10,000	5,000	5,000	2,500	2,000	1,000	500
1280	720	10,000	5,000	5,000	2,500	2,000	1,000	500
1600	900	10,000	5,000	5,000	2,500	2,000	1,000	500
1920	1080	10,000	5,000	5,000	2,500	2,000	1,000	500
2048	1152	10,000	5,000	5,000	2,500	2,000	1,000	500
4320	2432	10,000	5,000	2,500	1,250	2,000	1,000	500
8192	4608	10,000	5,000	2,500	1,250	2,000	1,000	500

Table 4.1: Number of iteration loops performed during power and performance analysis.

4.4.1 *Exploratory Runs and Fans*

Initially some exploratory runs with the software and power measurement facility were performed to highlight any issues that may arise. While running the tests, the cooling fan typically started immediately with neon and processor, when running the GPU, the fan started typically 1 second after the software started running, this is thought to be related to the 600MHz GPU frequency as opposed to the Processor 2.0GHz operating frequency.

For the largest images the fan could be heard and observed to be ramping up and down a range of speeds occasionally, especially when running the Neon processor, which could be observed as steps on the power graphs. The Linux fan driver for the Odroid XU4 is based on three temperature thresholds providing four fan speeds including ”off”. The system monitors the core temperature of the processor clusters to initially control the cooling fan and further apply DVFS to reduce core frequency and voltage if fan cooling isn’t successful. The initial runs with large images, 4K and above, also highlighted issues with the technique used to control fan speed and processor throttling with DVFS by the Linux system, generating power spikes greater than 20 Watts. Figs: 4.6 & 4.7 illustrate this effect. ARM DS-5 Streamline community edition was used to investigate the stepped variation of the Cortex A15 processor frequency from 2GHz downwards. This shows that the system then seemed to enter a cyclical sequence of, increasing fan speed to max, slowing down the processor via DVFS to cool it, processor cools - slow down the fans, re-increase the processor frequency for performance, increase fan speed etc. causing the large scale variations in power demand.

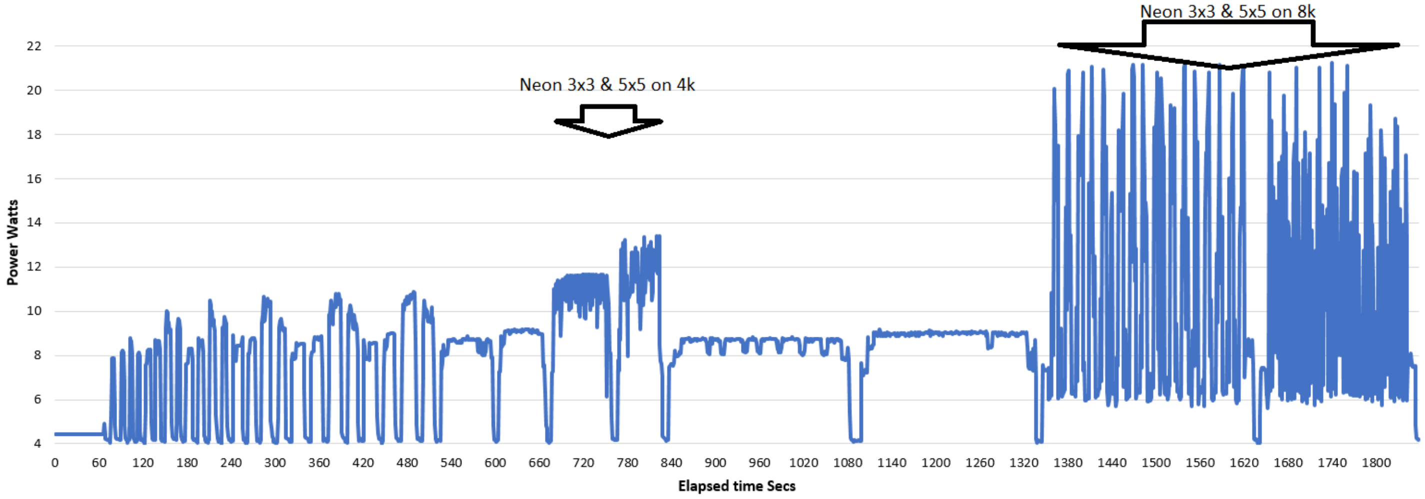


Figure 4.6: Plot of initial GPU and Neon Convolution runs showing neon power surging on 4k and 8k images.

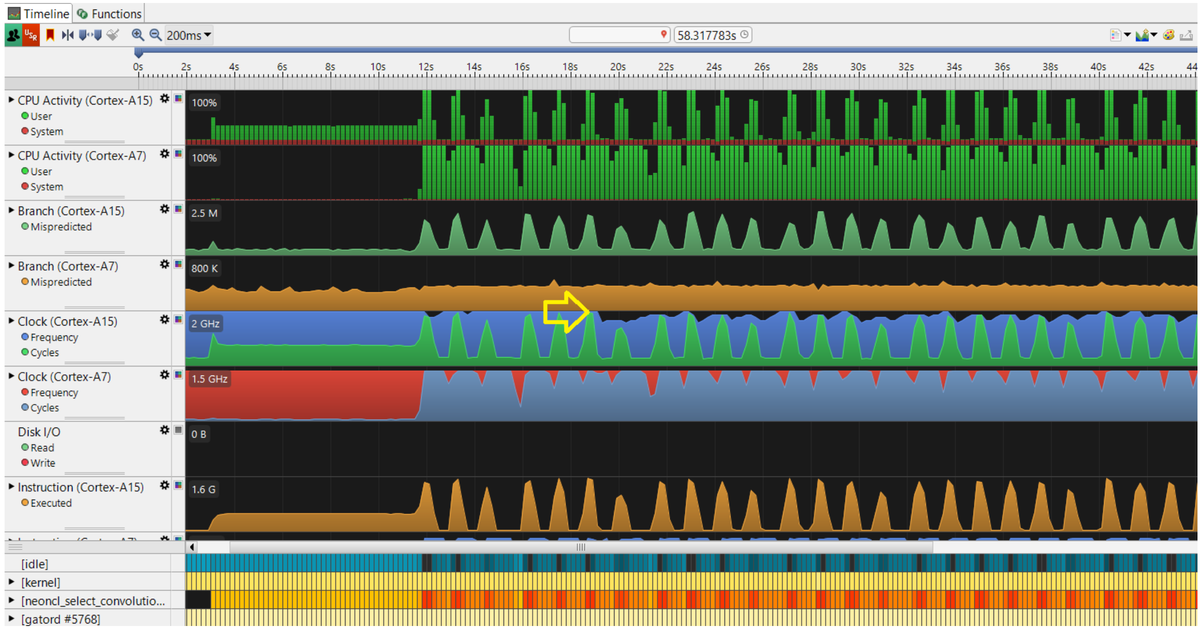


Figure 4.7: ARM DS-5 streamline view of DVFS application during 8k image, Neon 5x5 kernel. Yellow arrow shows start of cyclic DVFS.

This effect turned out to be due to having compiled the Compute Library version with debug enabled to gain access to some GPU registers via Streamline! The software was recompiled as a release version and the software and power consumption subsequently showed more reasonable behaviour having done so.
Observation 1: Care needs to be taken during intensive use of the Neon FPU (HD images continuously running) to avoid heating of the CPU and cyclic cooling by the OS.
Observation 2: Ensure software is compiled with debug off.

Because it appears the XU4, when idle, is running just below the fan switch on point. The XU4 was covered with a small box after which the fan started to run continuously and the static power, for fairness, with the fan running at first speed was determined to be the average at 5.4 Watts. This became the figure for subtraction of Linux Idle static power from the results, any further speed increases are considered as part of the run-time dynamic power consumption and should be included as part of run-time energy requirement.
Observation 3: XU4 Linux system idle power is to be taken as 5.4 watts.

The tests were then able to be run in three batches. The first run was to perform the openCV 1x1 convolution on all 9 image sizes, in increasing order, for reference. The second run was to perform OpenCV 3x3 followed by 5x5 convolution on all the image sizes, again in increasing order. The third run was to perform the Compute Library based 3x3 and 5x5 on the GPU, followed by 3x3 and 5x5 on the Neon processor, on the increasing image sizes. At the end of each set of runs the command line output log was saved for alignment and calculation of the required values in the spreadsheet.

4.5 RESULTS

The power measurement results from each set of runs were imported to a spreadsheet, an initial power plot was generated after each set of runs, to ensure it accurately resembled the plot on the TI EVM software. Three sets of power profiles for the runs were generated.

Fig:4.8, shows OpenCV 1x1 convolutions on the increasing range of image sizes ranging from 640 through to 8192 widths. One notable feature of this power profile is that the maximum power level, around 8 Watts is fairly constant through all the frame size tests. A fact to be borne in mind is that while the tests are running, Linux is running management tasks in the background, sometimes on other processors, explaining some of the spike and/or step anomalies.
Observation 4 Neon processing at higher definition for long time periods may show small steps and spikes due to OS events

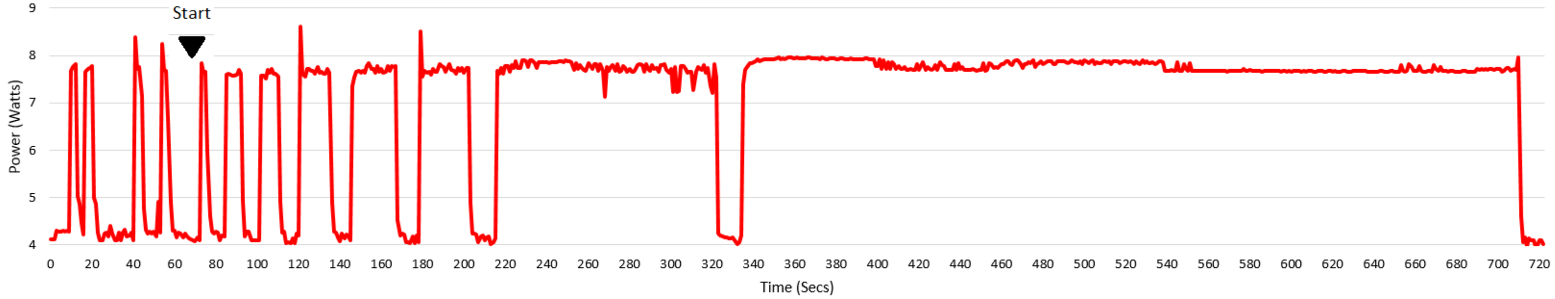


Figure 4.8: Power demand of OpenCV 1x1 convolution on increasing frame size.

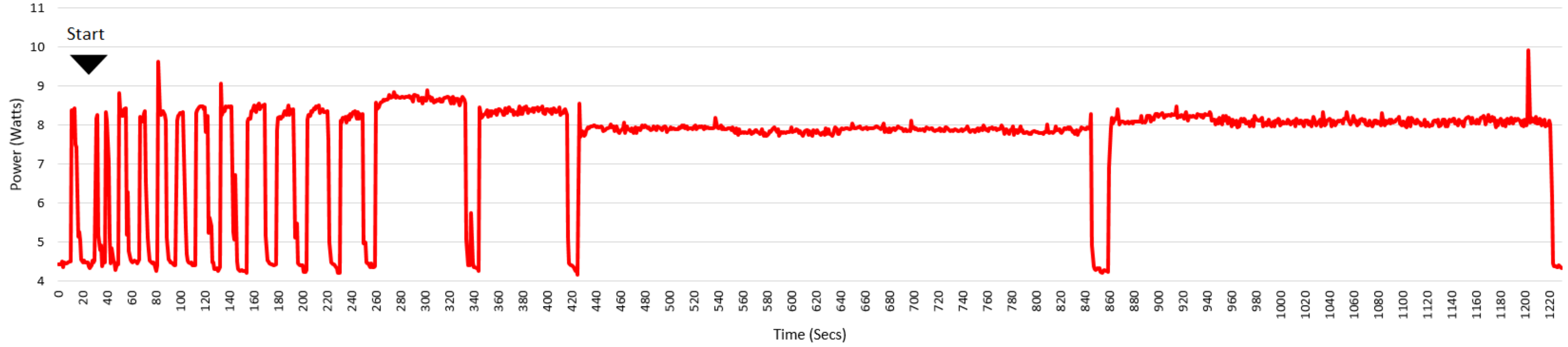


Figure 4.9: Power demand of OpenCV with alternate 3x3 & 5x5 convolution on increasing frame size.

Fig:4.9, shows OpenCV 3x3 followed by 5x5 on each of the increasing range of image sizes, so each image size shows a pair of 3x3 and 5x5 power pulses. Again the power profile shows a fairly constant maximum level response throughout the tests.

Fig:4.10, shows application of GPU 3x3 followed by GPU 5x5, Neon 3x3 then Neon 5x5 in that order on the increasing range of image sizes. At the end are the four 320x240 image power profiles. Below the main image is a magnified version of the two Neon 8k runs. So there are four power pulses for each image size. Note the obvious increase in power between the convolution size, GPU, Neon and with increasing image size. The upward steps in the increasing image sizes are due to the extra workload generating heat in the Exynos processor silicon, further requiring extra power to drive the higher fan speeds as the silicon temperature crosses the threshold settings in the Linux OS. The two right hand, Neon 3x3 and 5x5 on 4320 x 2432 and 8192 x 4608, appear noisier, this was due to the Linux OS cycling the fan and DVFS control loop albeit in a more controlled format and this may have skewed the results for that combination at the 4k and 8k image sizes. The two GPU 3x3 and 5x5 runs, at 8k image size, show small step intervals, this being due to the fans changing speed, which could be noticeably heard during the runs. It becomes noticeable at the long 8k runs that the GPU running at 600MHz is having little effect on the processor whereas the Neon processor which is an integral part of the CPU is having a thermal effect on the processor causing the fan/DVFS cooling to be cycled by the OS CPU management.

Observation 5: Neon FPU has a greater thermal effect on it's CPU.

Observation 6: GPU has no visible effect on CPU, only on fan speed.

The command window output information produced during the runs was coordinated with the spreadsheet power results to establish the start and end of the timed loops for each run. This is mainly required in the GPU run kernels due to the methodology in which OpenCL works. OpenCL has a one time, start-up and tear-down overhead associated with it during which the device, (GPU) characteristics are fetched and the hardware configured for the required task. Once the hardware is set up, no further action is required other than to execute the desired functions or kernels in this case.

The run time is used with a summation of the Power figures to generate an overall average power figure from which the Linux Idle Power figure is subtracted to provide an average kernel runtime power figure. This figure can then be divided by the number of iterations as previously indicated in Table: 4.1 in Section 4.4. This enables an overall calculation to generate the comparative frame energy and execution times and with further division by the frame pixel count to yield the pixel-kernel energy and execution time results which follow in summary form.

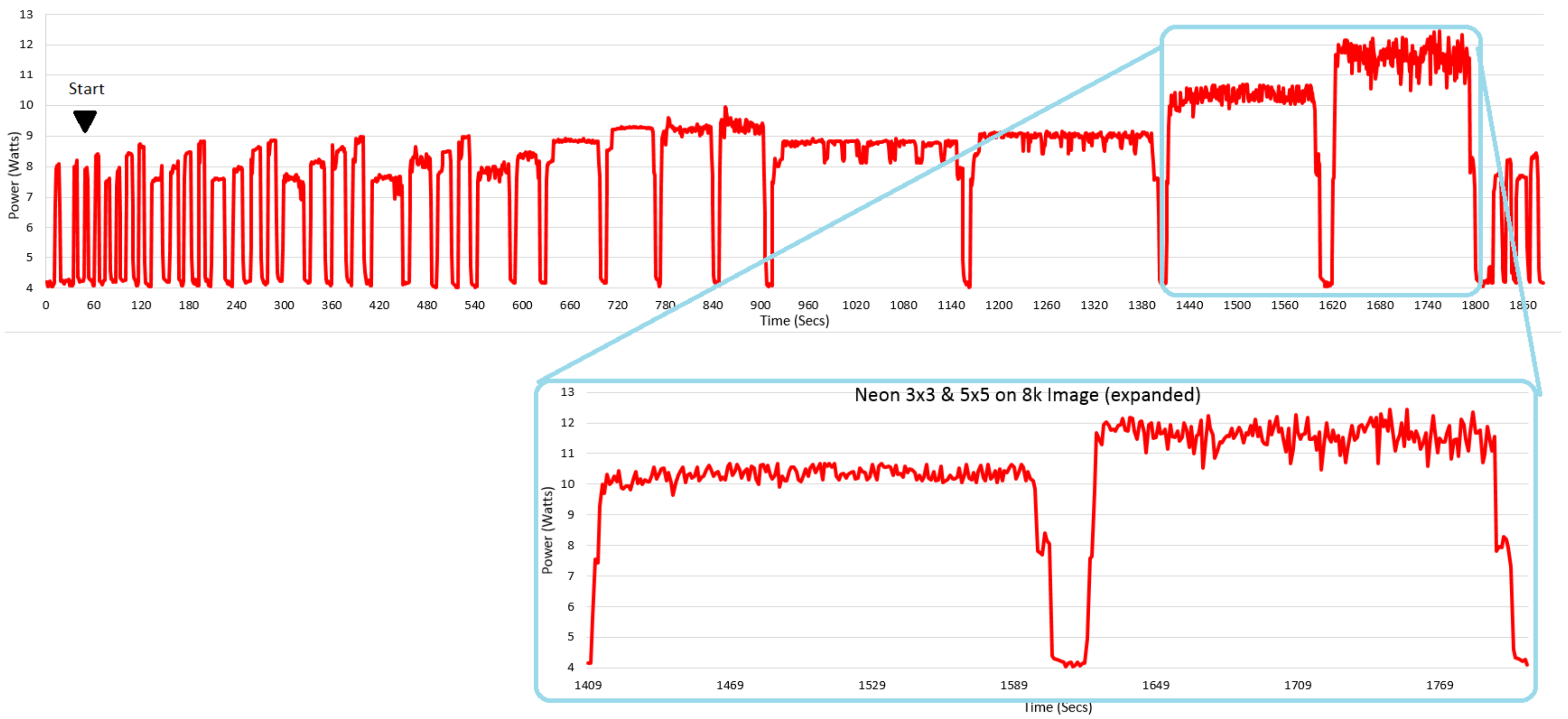
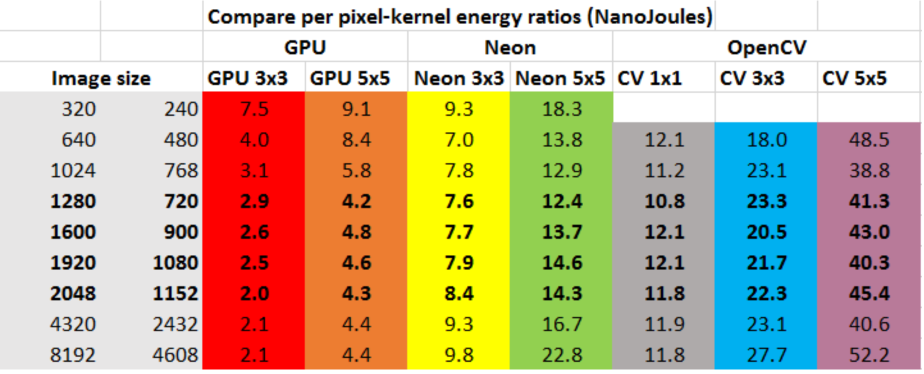


Figure 4.10: Upper: Power demand of GPU & Mali 3x3 & 5x5 convolution on increasing frame size. Lower: Expansion of the Neon 3x3 and 5x5 at 8k image size.

4.5.1 *Pixel-Kernel Results*

For illustrative purposes, the following result comparisons and presentations the tables and graphs have been reordered to show GPU, Neon then OpenCV result figures. Figs: 4.11 & 4.12 illustrate that for the GPU the advice given by ARM that GPUs only achieve efficiency in above HD image sizes and the higher energy figures reflect this, the 320 x 240 image size was included in the GPU test to reinforce this advice. The figures also demonstrate that the Mali T628 performs equally well illustrating consistency at all image sizes above 1k width whereas the Neon may be slightly more challenged, thermally and performance wise at 4K and above as witnessed by the increase in energy requirement. OpenCV seems to be slightly challenged perfor-



		Compare per pixel-kernel energy ratios (NanoJoules)						
		GPU		Neon		OpenCV		
Image size		GPU 3x3	GPU 5x5	Neon 3x3	Neon 5x5	CV 1x1	CV 3x3	CV 5x5
320	240	7.5	9.1	9.3	18.3			
640	480	4.0	8.4	7.0	13.8	12.1	18.0	48.5
1024	768	3.1	5.8	7.8	12.9	11.2	23.1	38.8
1280	**720**	**2.9**	**4.2**	**7.6**	**12.4**	**10.8**	**23.3**	**41.3**
1600	**900**	**2.6**	**4.8**	**7.7**	**13.7**	**12.1**	**20.5**	**43.0**
1920	**1080**	**2.5**	**4.6**	**7.9**	**14.6**	**12.1**	**21.7**	**40.3**
2048	**1152**	**2.0**	**4.3**	**8.4**	**14.3**	**11.8**	**22.3**	**45.4**
4320	2432	2.1	4.4	9.3	16.7	11.9	23.1	40.6
8192	4608	2.1	4.4	9.8	22.8	11.8	27.7	52.2

Figure 4.11: Comparison of energy (nanoJoules) per pixel-kernel per image size.

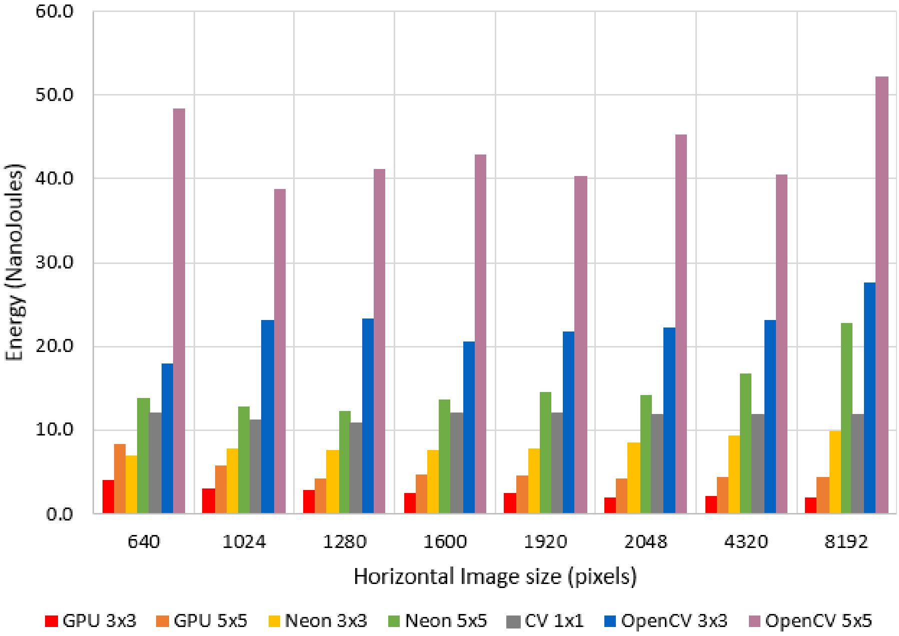
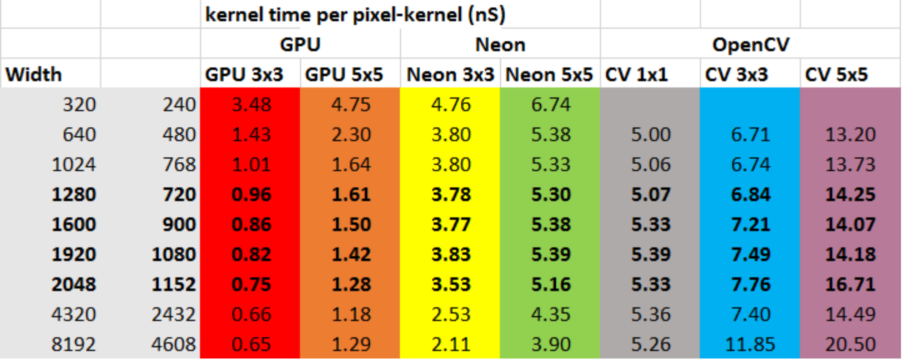


Figure 4.12: Energy requirement per pixel-kernel per image size.

mance wise at the 8k image size.

Observation 7: Only utilise XU4 T628 GPU with image sizes above 1024x768.

Observation 8: Only utilise XU4 Neon with images above 640x480 and ensure that above 4k running is restricted to intermittent operation.



		kernel time per pixel-kernel (nS)						
		GPU		Neon		OpenCV		
Width		GPU 3x3	GPU 5x5	Neon 3x3	Neon 5x5	CV 1x1	CV 3x3	CV 5x5
320	240	3.48	4.75	4.76	6.74			
640	480	1.43	2.30	3.80	5.38	5.00	6.71	13.20
1024	768	1.01	1.64	3.80	5.33	5.06	6.74	13.73
1280	**720**	**0.96**	**1.61**	**3.78**	**5.30**	**5.07**	**6.84**	**14.25**
1600	**900**	**0.86**	**1.50**	**3.77**	**5.38**	**5.33**	**7.21**	**14.07**
1920	**1080**	**0.82**	**1.42**	**3.83**	**5.39**	**5.39**	**7.49**	**14.18**
2048	**1152**	**0.75**	**1.28**	**3.53**	**5.16**	**5.33**	**7.76**	**16.71**
4320	2432	0.66	1.18	2.53	4.35	5.36	7.40	14.49
8192	4608	0.65	1.29	2.11	3.90	5.26	11.85	20.50

Figure 4.13: Comparison of time (in nanoSeconds) per pixel-kernel per image size.

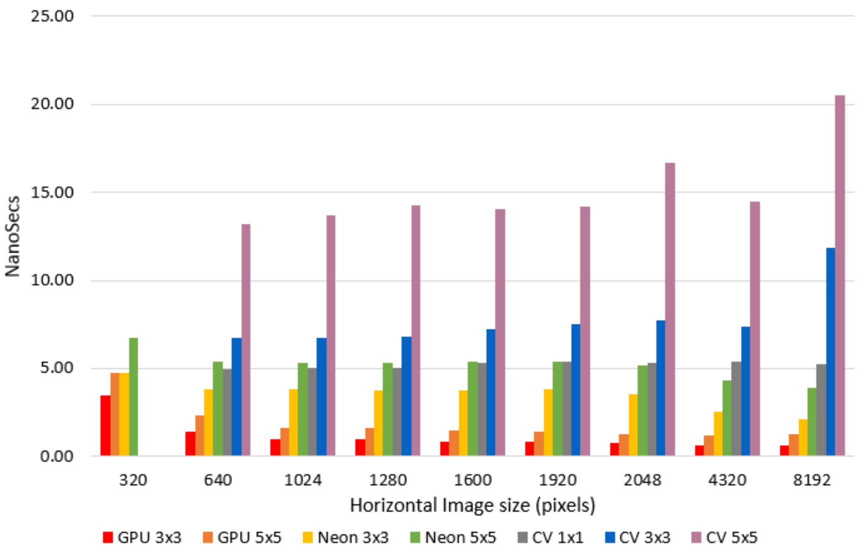
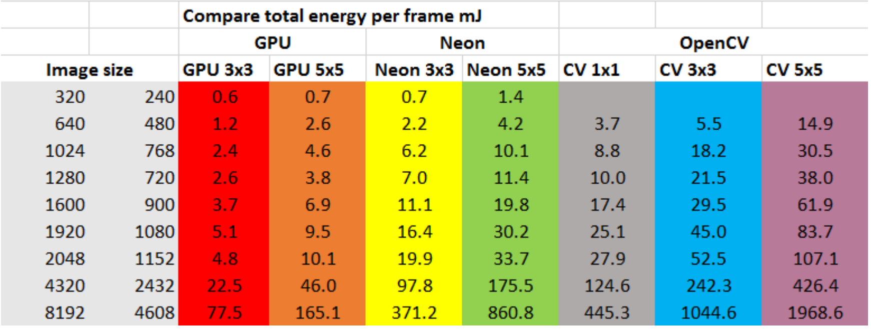


Figure 4.14: Compute time requirement per pixel-kernel per image size.

Figs: 4.13 & 4.14 similarly show that performance of the GPU at low resolution should obviate it's use for small images and that there is a consistent performance across images > 1k width. The Neon appears to execute faster at the 8k image size while the power increases as pointed out previously, certainly the right hand side of Fig: 4.10 shows noise at the top of the power waveform, at the higher current which is due to the cycling effect of fan speed variation and DVFS, each between their lower and higher parameters of operation. Our opinion is that use of Neon alone to continuously process these large images, with the Exynos 5422 technology, may be working beyond its' capability.

4.5.2 *Frame Energy and Performance Results*



		Compare total energy per frame mJ						
		GPU		Neon			OpenCV	
Image size		GPU 3x3	GPU 5x5	Neon 3x3	Neon 5x5	CV 1x1	CV 3x3	CV 5x5
320	240	0.6	0.7	0.7	1.4			
640	480	1.2	2.6	2.2	4.2	3.7	5.5	14.9
1024	768	2.4	4.6	6.2	10.1	8.8	18.2	30.5
1280	720	2.6	3.8	7.0	11.4	10.0	21.5	38.0
1600	900	3.7	6.9	11.1	19.8	17.4	29.5	61.9
1920	1080	5.1	9.5	16.4	30.2	25.1	45.0	83.7
2048	1152	4.8	10.1	19.9	33.7	27.9	52.5	107.1
4320	2432	22.5	46.0	97.8	175.5	124.6	242.3	426.4
8192	4608	77.5	165.1	371.2	860.8	445.3	1044.6	1968.6

Figure 4.15: Comparison of energy (Joules) per frame per image size.

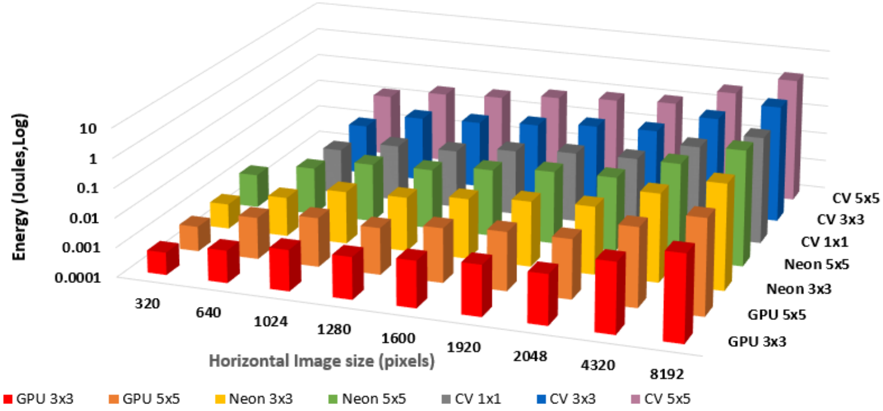
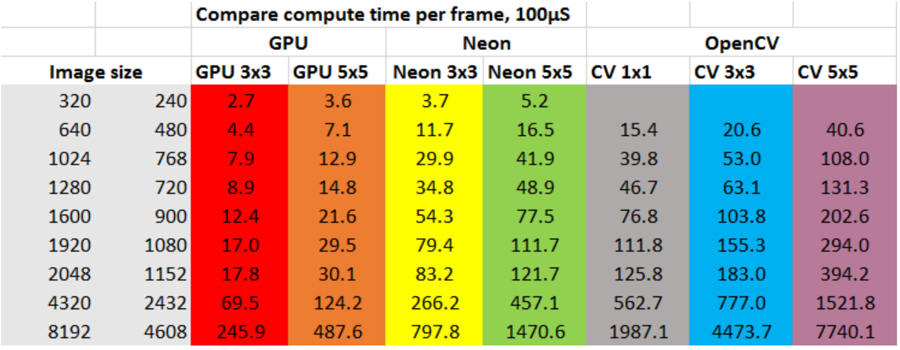


Figure 4.16: Logarithmic view of energy per frame per image size.

Figs: 4.15 & 4.16 show the energy demand of each frame processed across each image size, the vertical axis of the plot is scaled logarithmically to demonstrate the exponential impact of image size on energy consumption as estimated in Chapter: 2 Table: 2.1. The results reinforce the ability of the GPU to efficiently reduce energy consumption, by roughly an order of magnitude, in continuous whole frame image processing. The Neon processor also helps to reduce energy to around 40% of the processor consumption at the <4k image size. It has to be borne in mind that the figures were derived from continuous frame processing which generates significant heat and higher fan speeds so at reasonable frame rates, < 30fps, may well give more consistent results for larger images.

Figs: 4.17 & 4.18 show the frame execution times across the image sizes, the vertical axis of the plot is also scaled logarithmically and



		Compare compute time per frame, 100µS						
		GPU		Neon			OpenCV	
Image size		GPU 3x3	GPU 5x5	Neon 3x3	Neon 5x5	CV 1x1	CV 3x3	CV 5x5
320	240	2.7	3.6	3.7	5.2			
640	480	4.4	7.1	11.7	16.5	15.4	20.6	40.6
1024	768	7.9	12.9	29.9	41.9	39.8	53.0	108.0
1280	720	8.9	14.8	34.8	48.9	46.7	63.1	131.3
1600	900	12.4	21.6	54.3	77.5	76.8	103.8	202.6
1920	1080	17.0	29.5	79.4	111.7	111.8	155.3	294.0
2048	1152	17.8	30.1	83.2	121.7	125.8	183.0	394.2
4320	2432	69.5	124.2	266.2	457.1	562.7	777.0	1521.8
8192	4608	245.9	487.6	797.8	1470.6	1987.1	4473.7	7740.1

Figure 4.17: Comparison of time (100 uSecs) per frame per image size.

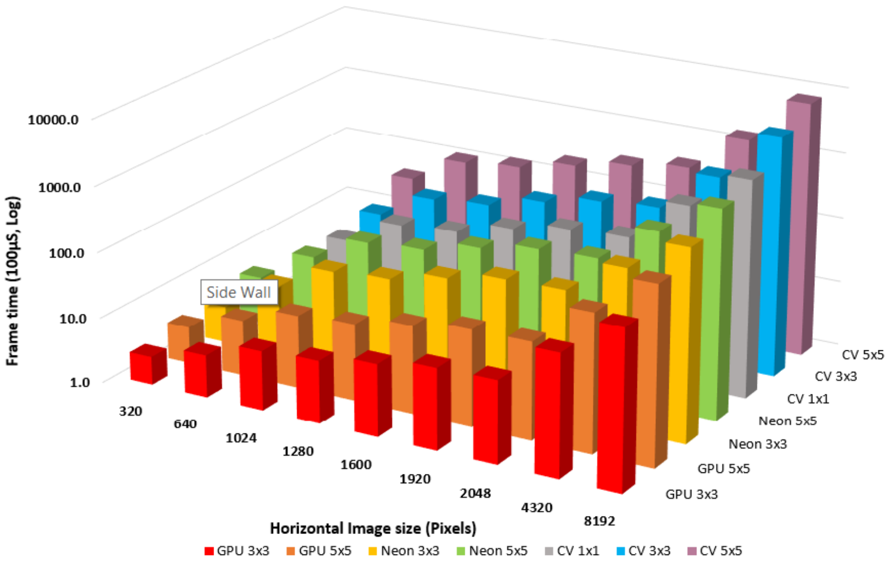


Figure 4.18: Logarithmic view of time per frame per image size.

likewise demonstrates the exponential increase in processing times as also estimated in Chapter: 2 Table: 2.1. This demonstrates that for 8k images, significantly higher performance SoC components are required as the execution time for the frame at 245mS would only render around 4 fps video update rate, unless we improve performance by utilising approximation methodology.

4.6 SUMMARY

The results demonstrate that, while GPUs and Neon DSPs may appear to show a higher wattage while running, in actual fact these devices are far more efficient energy-wise than running the same function on a processor with the GPU emerging with the greatest efficiency and performance overall, dispelling the often held conception that GPUs are energy-hungry devices.

These results reinforce the concept of being able to perform a significance based on two thresholds in an image to yield three levels of processing, in parallel form, such that most significant areas could be computed with 5x5 convolution performed on the GPU, mid significant areas with 3x3 performed on Neon DSPs and low significance areas with 1x1 by the processor, all of which will be discussed in the next chapter.

5 SIGNIFICANCE DRIVEN ADAPTIVE APPROXIMATE COMPUTING

In this chapter we demonstrate a novel idea of utilising image significance to identify the more significant areas in an image. By utilising a dynamic dual threshold system we can ultimately partition these image areas for three different levels of image processing, based on a percentage requirement by the user, ranging from no processing in insignificant areas through to more complex or accurate in high significance areas. We then go on to demonstrate that, due to the lower requirements for processing effort which generates slack or idle time, we can manage the CPU frequency and voltage to further reduce computational effort and power by judicious use of this slack time. We term this technique as significance driven adaptive approximate processing. In Section 5.1 we will introduce the idea of how we will utilise approximate image significance to direct areas of the image to a level of processing relevant to that significance. Section 5.2 introduces the rationale of a three level processing derived from two adjustable thresholds determine by the user requirements of percentages of the whole image to be processed at these levels. Section 5.3 explains how we utilise a series of stepped percentage levels that trade off power, performance and quality to achieve a demonstrable computational and power efficiency at the quality expense of not processing the insignificant areas. Section 5.4 will describe the approach used to design the operation of the demonstrator along with the reduced size significance and pooled matrices, used to determine which candidate area is selected for the relevant processing. Section 5.5 explains in detail the runtime option selection and operation of the software demonstrator, along with evaluation of the results. Section 5.6 gives a background of the challenges discovered that prevented the demonstrator moving on to a parallel approach or implementation of Neon or GPUs in the image processing. This section goes on further to indicate the efficiency savings that could potentially be achieved over OpenCV if the Neon/GPU phase could be implemented. Section 5.7 summarises this chapter.

5.1 INTRODUCTION

Current image processing commonly utilises 3 x 3 and/or 5 x 5 convolutions across every pixel in an image. It is unusual to witness the use of 7 x 7 or 9 x 9 convolutions due to the extra overheads of energy and performance at the processor level.

The ability to detect significance in an image as outlined in Chapter 3, along with evidence gathered from the case study based on image significance generated a single threshold approach, dividing the image into areas, below threshold, that were processed as a 1 x 1 or 3 x 3 kernel and areas that were processed, above threshold, with 3 x 3 or more precise 5 x 5 convolutions. The Odroid XU4 was used for this work and OpenCL was utilised to generate the kernel convolution in a much quicker time than it's software CPU equivalent, via the Mali T628 GPU.

In this work, reference may be made to a 1 x 1 convolution as the data below threshold is unprocessed or the equivalent of a 1 x 1 convolution with unity gain. This reference may also be seen in relation to Neural Network processing but in that case the 1 x 1 represents a matrix operation on multi-dimensional Tensors.

For a practical demonstrator this has been limited to a two threshold system giving three levels of processing as follows:

- 0 Below threshold, simply pass data across without computation equivalent to a 1 x 1 convolution with unity value.
- 1 Above 1st threshold, perform 3 x 3 convolutions
- 2 Above upper 2nd threshold perform 5 x 5 convolution in those areas.

It can easily be envisaged that extra thresholds could be added to provide 7 x 7 and 9 x 9 convolutions, dual combinations of 3x3 and 5x5, or even offering more precise higher order convolutions, in highly significant areas of an image. The software demonstrator was designed with expansion of this concept borne in mind.

5.2 RATIONALE AND RELEVANT RESEARCH

In Image Processing with OpenCV, currently kernels are applied across whole image. Significance creates an opportunity to only process significant areas and leave insignificant areas with no processing. The case study investigated this concept using the Arm Mali development kit version of OpenCL with a single threshold level so that the more significant areas in an image were processed with 3x3 kernels and below threshold with 1x1, effectively no processing. A further variation was to use 5x5 above the threshold and 3x3 below threshold.
In this demonstrator we illustrate usage of two thresholds providing three levels of processing at 1x1, 3x3 and 5x5 convolutions for each of the workgroups. The demonstrator software could be easily altered to

implement more threshold levels, other convolutions or combinations of convolutions.

A GPU enforces a minimum work-group size, dependent on GPU architecture but in the case of the XU4 T628 GPU this appeared as 64x64 pixels. This work-group size was adopted for this work, while always bearing in mind the ability to process other sizes for different GPUs.

The work-group size imposes another constraint when working with OpenCV, having to add padding at the edges or cropping the image in order to make the source image an integral number of work-group sizes. With Arm Compute Library the GPU takes care of this issue.

The structure of the demonstrator was architectured to utilise approximate significance to perform adaptive filtering based around the OpenCL methodology of utilising Workgroups and submitting them for GPU processing by utilising threads programming, Lawlor [84], Connors [85]. The functionality of the convolution filtering section, in an OpenCV version, attempts to follow the techniques employed in the Arm Compute Library which utilise the Neon FPU or GPU, with the intention of ultimately utilising the Compute Library methods in order to provide an easier migration path to a Neon/GPU version.

Relevant research that addresses the issue of identifying significance to process in images, tend to utilise compute intensive techniques, include: Statistical significance of features in digital images is addressed by Godtliebsen [56] which utilises local probability distributions and gradients around pixels to locate significant areas in noisy medical images. SEEDS by Van-den-Bergh [35] approach uses an iterative hill climbing technique to locate and process significance. A spectral residual technique discusses that natural images have highly predictable distributions and applies a compute intensive saliency technique, Hou [41]. Image database retrieval that approaches human perception techniques with the aid of tetrominoes and tetrolet transforms by Raghuwanshi [66, 67]. A more recent development by Metwalli [86] is a proposal for static significance analysis to identify areas of a programme in which approximation should be applied.

5.3 POWER, PERFORMANCE, QUALITY TRADE-OFF MODELS

For the proposed three level of interest system we require two thresholds. Approximate significance along with the idea of multiple variable thresholds can be easily achieved at the demonstrator level by the use of threshold slider bars to control the significance interest levels which would then allow fine control of both thresholds over the full pixel value ranges with the constraint that the higher level (threshold2) must always be greater than the lower (threshold1). As the threshold levels are raised from zero, fewer work-groups are processed in each of the two higher significant areas reducing processing of the more complex convolutions which in turn will reduce energy demands and will also affect the quality of the resultant image. This variable threshold ability can provide a useful control knob for interactive or Machine Learning instances of image processing.

For the demonstrator it was decided to have a handful of preset Power, Performance and Quality, PPQ, levels in order to enable a flexible, easier set-up during experiments with varying pixel size images, videos and camera input. These threshold levels are dynamic, based on the demanded percentages of the image to be processed at each level. The two threshold levels are determined by an interactive process based on a histogram of the significance levels of the current frame, Fig: 5.1 shows a simplified flowchart version, more detail of this interactive process will be provided later in Section: 5.5.2.

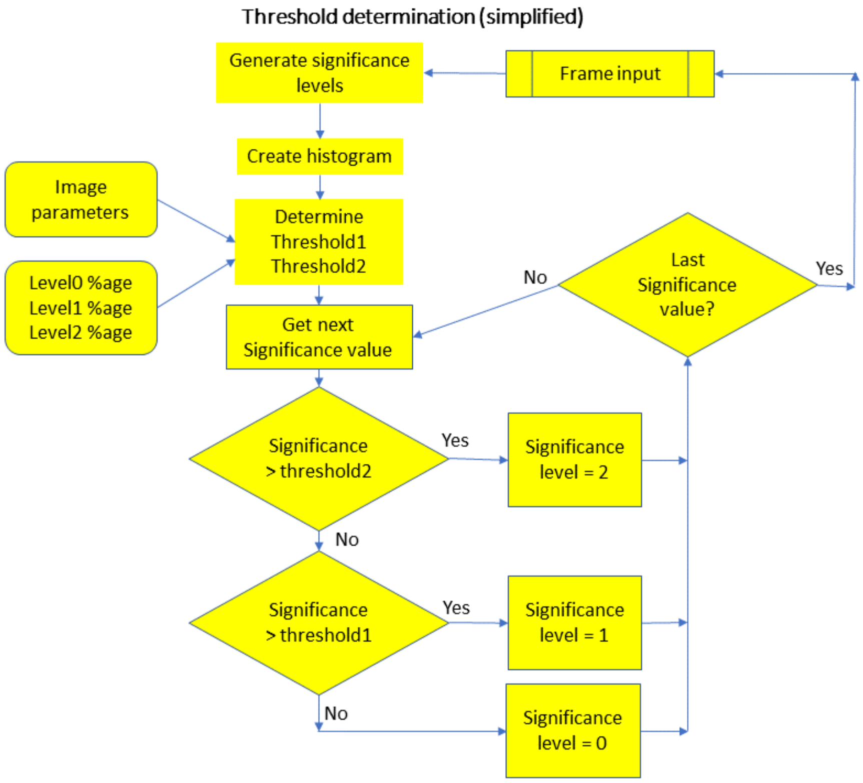

Threshold determination (simplified)
Generate significance levels
Frame input
Create histogram
Image parameters
Determine Threshold1 Threshold2
Level0 %age Level1 %age Level2 %age
Get next Significance value
No
Last Significance value?
Yes
Significance > threshold2
Yes
Significance level = 2
No
Significance > threshold1
Yes
Significance level = 1
No
Significance level = 0

Figure 5.1: Determining interactive threshold from image significance.

The updated thresholds are then used to segregate the workgroups into their determined levels and designated to their equivalent processing. In this demonstrator, below threshold1, (ie level0), a 1x1 is performed. Between threshold1 and threshold2, (level1), a 3x3 is performed. Above threshold2, (level2), a 5x5 is performed. Table 5.1 shows the preset levels. As an example PPQ level 0 will effectively ignore

90% of the image workgroups that are below threshold1, process 5% of the image with a 3x3 convolution that lies between threshold1 and threshold2, the remaining 5%, above threshold2 are processed with 5x5 convolution. PPQ levels 5 and 6 are effectively a whole frame 3x3 or 5x5 process, purely for comparison demonstrations.

PPQ levels for Demonstrator software			
	Level 0	Level 1	Level 2
PPQLevel	%age	%age	%age
0	90	5	5
1	80	10	10
2	70	15	15
3	60	20	20
4	50	30	20
5	0	100	0
6	0	0	100

Table 5.1: Percentage proportions of image significance assigned to processing level by PPQ value.

5.4 PROPOSED ADAPTIVE APPROXIMATE SIGNIFICANCE CONCEPT

The concept was envisaged as a multi step process.

1. Calculate Approximate significance of the image, generating a smaller Approximate significance matrix, smaller Matrix access requires less energy and execution time.

2. Pool the Significance matrix by using Mean, Maximum value or Dynamic range of all Significance in each of the the equivalent work-groups.

3. Generate target work-group numbers to be processed at each level based on a percentage of the work-area size.

4. Generate a significance Pooled matrix indicating to which level each work-group should be processed, 0,1,2.

5. Compute the convolution of each workgroup based on the pooled significance value being below the first threshold, 1x1, between the low and higher threshold, 3x3, or above the upper threshold, 5x5.

5.4.1 *Structure and Operation of Software Demonstration Model*

The software was written, based on the earlier approximation software with a number of extra features highlighted by experimentation with the earlier version, thereby enabling more flexible experimentation with Larger images. A command line option system along with runtime slider controls were utilised to provide the following options:

- A choice of image source of either a video camera input, video file input or still image input which is recalculated on a 10fps basis. The latter being useful for debugging and performance checking.
- Choice of 2x2, 4x4, 8x8 or 16x16 sub arrays for calculation of the approximate significance, in order to allow a greater reduction in the significance arrays for higher definition images above 2k width eg a 2560 x 1600 image will be reduced to a 640 x 400 significance array with a 4x4 sub array and 320 x 200 with 8x8, offering better performance for larger images.
- Reduce the significance values matrix dimensions. In the original demo software this matrix was the same size as the source image in order to act as a thresholded image mask for the early research explorations. By reducing this to one matrix value per sub-array, 2x2, 4x4, 8x8 or 16x16, each dimension is reduced by the array size, providing faster access to values while accessing significance values in later computations.
- Choice of workgroup size 16x16, 32x32 or 64x64. The larger 64x64 matches the GPU based ARM Compute Library workgroup size and in actual fact provides faster computation times.
- A Pooling or clustering of the approximate significance values, corresponding to the workgroup area, into a classification level for the two level thresholding of workgroup values based on mean, maximum value or dynamic range (Maximum - minimum).
- Introduction of upper and lower threshold sliders to enable division of the Pooled matrix into 3 areas of significance, 0 for low significance, 1 for mid and 2 for high significance. This feature allows each Workgroup to be selected for dedicated processing relative to its significance. Fig: 5.2 illustrates the relationship between the matrix sizes and the significance processing.
- Histograms of these pooled values are utilised to automatically set the threshold level sliders used to determine what processing is utilised for each Workgroup significance value.

- While the demo is running a PPQ slider was introduced to enable five preset percentage levels to be used to generate target counts for the three processing levels for Power, Performance and Quality, 0 being the lowest Power and quality and 4 being a higher level of Quality and Performance at the expense of Power consumption. The percentage levels are converted to workgroup proportions dependent on the size of the image. By defining two percentage values for the 5x5 and 3x3 computation the third is derived and used to represent control knobs that can be applied to the image. A further two levels 5 and 6 provide 100% 3x3 or 5x5 processing of the whole image, merely for comparison purposes. The previous Table: 5.1 shows percentage levels for the seven slider positions, 0 to 6.

- When the Software is run on the Odroid XU4 platform, there is a facility to utilise DVFS to alter the CPU core frequency and voltage by a system call. The runtime software utilises a slider "cpu-freq" offering 19 positional step settings that vary the CPU frequency and/or voltage, according to the Samsung Exynos 5422 specification and controlled by the device tree settings in linux, [87], 18 representing 2GHz down to 0 representing 200MHz, in 100MHz steps. Thereby allowing reduced frequency and energy running when lower values of PPQ are selected and sufficient slack time is available.

- The introduction of a 64x64 workgroup can cause issues with border adjustment. If the image dimensions, height and width, are not exactly divisible by the workgroup edge size then extra border rows or columns have to be added in order to prevent exceptions caused by the OpenCV dimension checking. The ARM Compute Library has built in functions to deal with this and padding is handled during the allocation functor but when using OpenCV care must be taken to handle and adjust image dimensions by padding the image with borders before calling kernel functors. This is performed by a method in the software.

- An option to view extra windows while debugging Pooling etc.

- An option to remove all windows other than the slider during performance testing.

In the following explanations the OpenCL concept of work-area, work-group and work-item have been adopted. Referring to Fig: 5.2 Work-area represents the total WXQGA image size of 2560 x 1600 pixels. The 4x4 approximate significance sub-arrays can be considered as work-items. The image in this example is split into 64 x 64 pixel Work-groups each containing 16x16 work-items with the Pooled significance level of each group held in a 40 x 25 pooled matrix. The array of pixels in the 64x64 work-group are processed as a number of work-items that are normally processed in parallel and is dependent on the structure of the parallel array.

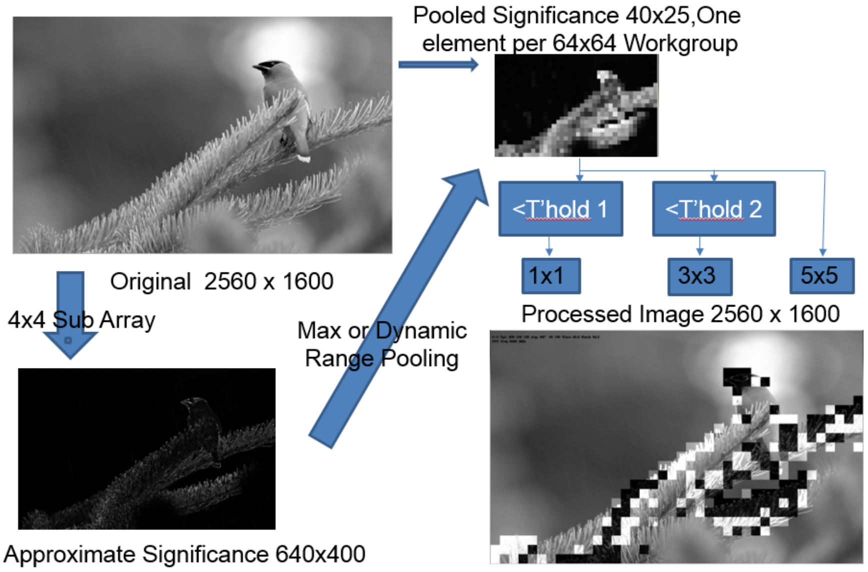

Pooled Significance 40x25,One element per 64x64 Workgroup
<T'hold 1
<T'hold 2
1x1
3x3
5x5
Processed Image 2560 x 1600
Original 2560 x 1600
4x4 Sub Array
Max or Dynamic Range Pooling
Approximate Significance 640x400

Figure 5.2: Relationship of image matrices, sub-arrays and workgroup in image processing demonstrator software.

In design of all aspects the software, care was taken to utilise integer powers of two in order to utilise left and right shifts to replace multiplication and division operations.

5.5 Software Demonstrator Development and Operation

The software was developed to run on Ubuntu Linux based x86 and ARM platforms based on the earlier Image Significance Demonstrator in Chapter: 3 and improving on some of the lessons learned during that phase, along with the thresholding features exposed during the Case study. It was anticipated that the software could explore utilisation of the GPU, Neon DSP or Thread implementation on the Multi-processor availability, to perform the convolutions but a number of issues, outlined in Section: 5.6, prevented either of these desires to occur therefore development proceeded as a software model only.

For this section, a link to the OpenCV demonstrator software can be found on GitHub via the web link in the following reference, [88] (Last checked 1 Oct 2019).

5.5.1 *Runtime Option Selection*

The application is called from the Linux command line with ./SigApproxCamCap utilising the <unistd.h> getopt() interface [89] (Last checked 1 Oct 2019). Help on the command options available can be found by entering ./SigApproxCamCap -h at the command line. which produces the output shown in Table: 5.2.

```
Print this help info
Call with selections of following arguments
-i image <filepath/filename>Path & alternate file to <default>
-v video <filepath/filename>file sequence for input
-c Use camera input instead of file
-t (two) 2x2 sub array size
-q Quad 4x4 sub array size
-o Octal 8x8 sub array size
-x (hex) 16x16 sub array size
-s small WorkGroup size 16x16
-m medium WorkGroup size 32x32
-l Large WorkGroup size 64x64
-w Use extra windows for debug of Poolin
-n No windows for performance testing
-M Significance based on max value in Workgroup
-d Significance based on dynamic range value in Workgroup
Default run action is -cqlM
```

Table 5.2: Software demo help output

Command line input can include a combination of commands but there are some mutual exclusions that must be observed which are explained as follows:

The first three i,v and c set the source of the image to be processed and only one must be selected. 'i' and 'v' expect a file path to be provided to access an image file,'i' or video file,'v'. 'c' utilises an attached video camera for the image source.

The next four arguments, 't' (2x2), 'q' (4x4), 'o'(8x8) and 'x'(16x16) of which only one must be selected, set the sub-array size from which one pixel value is extracted for the Approximate Significance calculation. There is an associated fixed offset value to select the pixel value just above and to the left of each of the sub-array centres, e.g. (1:1) for a 4x4 sub-array. This fixed option could easily be changed to a selectable option if ever required. These choices are provided to enable experimentation with larger sub-arrays for higher resolution images which will reduce the computation required for such images.

The work-group arguments 's' (16x16), 'm' (32x32) and 'l' (64x64) of which only one must be selected, were provided to allow for work-group size comparison to match those that may be selected when using a GPU. Early experimentation with the OpenCV based software has shown that the extra context switching overhead of the extra number of work-groups generated by the two smaller sizes slows the computation significantly.

The Windows commands 'w' and 'n' are exclusive as there is no point in providing extra debug windows while switching window display off altogether.

The Pooling selections 'M' (maximum) and 'd' (Dynamic Range ie Max - Min) are essentially a choice of one or the other.

The command line options, when processed set up the runtime variables and a structure of inter-related variables is used to deal with adjustment and correlation between sizes of the full image, the Approximate Significance matrix and the Pooled Significance, work-group based, Matrix.

5.5.2 *Runtime Operation*

The software now opens up the selected image source, camera, video file or still image file. The still image file is treated like a video file with the image being reprocessed on a 10 frames per second basis, this allows debug testing of a stable image to ensure the processing is consistent and to experiment with larger images, >2k pixels width.

The software then sets up a selection of display windows, dependent on the command line options to show the results of the processing. One further window with sliders is set up in order to control the processing and display the adaptive thresholding.

The image structure variables are then initialised to suit the image size and the subsequent sizes of the Approximate Significance and Pool matrices dependent on the command line selections.

As mentioned previously, the source image dimensions, height and width, need to be checked that they are evenly divisible by the work-group size, if not extra padding needs to be added to both edges of the non compliant dimension(s) and the image dimensions, in the structure, updated accordingly. The matrices to be used in the significance processing can now be defined and dimensioned as follows:

- Approximate Significance matrix, approxSig size = (Source size) / (Sub-Array Size).
- Destination Matrix for processed image, dst, Same size as Source.
- Pool values Matrix, (Pool size) = (Source size) / (Work-group size) holds Pooled significance values, Maximum or dynamic range.
- Thresholded level Pool values matrix, sigPool, is the same size as Pool but holds the thresholded level values, 0 to 2, relative to Pool value and threshold levels.

After a bit of housekeeping the main processing loop is now entered in an infinite loop.

In the absence of C++11 support by OpenCV 3.3, the chrono library could not be used so a self created timing system was used to generate a 1/fps frame timer update flag and a one second timer flag.

The one second timer flag is used to update the target threshold values, adjust_thresholds(), based on the selected PPQ value which defines the percentage of Work-groups to be processed at each threshold level. This process is complicated by the fact that the calculated threshold can vary dramatically between frames when processing video as it is derived from a count of the number of Workgroups processed at each of the three levels of processing, 5x5, 3x3 and 1x1, so the counts are averaged over a number of frames, typically 4 or 8, in order to bring some stability to the overall process of the automatic update mode. The threshold values are derived from summing the histogram of the values in the Pool matrix and is calculated to be the nearest histogram index required to achieve the required threshold targets. In order to minimise the calculation of the threshold level the target levels are calculated by counting from 255 down to 0 as the higher two threshold are generally smaller than the level 0 count. So the percentage levels are converted to a work-group count for the two upper threshold levels, 2 & 1, and the remaining values are therefore level 0.

If the PPQ slider is changed at any time a semaphore is set for the software to call a routine, changePPQvars(), to convert the 3 preset percentage levels to target level work-group counts, for use in adjusting the thresholding of the significance data.

The frame timer function sets a semaphore for the software to retrieve the next video frame when the frame interval time has expired. The inter-frame time interval is created from the fps rate recovered from the camera or the video file metadata. In the case of still images a preset 10 frames per second rate is used in order to offer sufficient time to process large images > 2k width during large image investigation.

The software, if border padding has been earlier identified, then has to adjust each frame image before video processing, camera or file, takes place.

The next step is to calculate The Approximate Absolute Deviation values for the current frame, storing the resultant values in the ApproxSig matrix.

Having updated this matrix, the next task is to update the Pool and sigPool values which represents another level of complexity in the processing due to the variable sub-array sizes and work-group sizes. The ApproxSig Matrix dimensions are the source image size divided by the sub array size and the two Pool matrices dimensions are the image size divided by the work-group size. So the Pooled values are derived from an equivalent positional rectangular group of ApproxSig values of size: ApproxSig size divided by the Pool size.

The following worked example, using a convention for array sizes of Mat(width,height), shows the calculation of the matrix element sizes:

- Original image Source(2560,1600)/Sub-array(4,4) yielding a matrix ApproxSig(640,400)
- Original image Source(2560,1600)/Work-Group(64,64) yielding a matrix Pool(40, 25)
- ApproxSig(640,400)/Pool(40,25) yielding a pooling matrix Rectangle(16,16)

So our example shows that each Pool element is derived, in this case, from a 16x16 rectangle of significance values, the derivation being based on The maximum value in that rectangle or the Dynamic range, (max-min) the value of which is entered into the Pool matrix at it's relevant position. During this processing phase the Pool values are tested against the Upper and lower thresholds in ASPool() to classify that value as 2 if above upper threshold, 1 if above lower threshold, and 0 otherwise, below lower threshold by default. So this significance level is entered into the sigPool Matrix, ready for the final processing stage.

The software now performs, via ComputeASperSig(), one of three Image Processing kernel convolutions on the source gray image utilising the OpenCV Rect class to select each rectangular area work-group in turn and perform the appropriate 1x1, 3x3 or 5x5 convolution as per the work-groups classification in sigPool and store the result in equivalent image area in the destination matrix. The filters shown in the Demonstrator are purely chosen and adapted to illustrate where processing has taken place, so the image appears the same as the source where level 0, 1x1, convolution has occurred, lighter shade of grey where level 1, 3x3 sharpening filter, has occurred and a darker shade where the Sobel, 5x5, filter has taken place.

The software creates tell-backs in order to monitor performance in real time. These tell-backs appear in two places, the destination frame window and the Command line terminal. The frame window values are updated every frame period and the command line terminal values are updated once every second. The displayed information comprises:

- Target level 0-2 workgroup counts which are the proportions requested via the percentage values selected by the PPQ slider value 0 to 4.
- Average achieved workgroup counts at each level which are the nearest values that can be achieved to the target based on the sigPool value histogram.
- The processing time for the image measured from just after the timers have been set up and before the image dimensions are

adjusted to the final image processing after the image processing has taken place.

- The slack time is also displayed, derived from the frame time minus the image processing time.
- In addition, the destination image displays an extra line with the CPU core frequency, only utilised on the Odroid XU4 and targeted at the exynos 5422 processor, this shows the current CPU frequency in conjunction with the cpu frequency slider in the sliders window.

The software displays the live RGB image along with the grey source, slider and processed destination windows. The command line options discussed previously can be used to display other windows for additional information or debug. Figure 5.3 shows a frame from a video of an overflying skein of geese, the geese are barely viewable in the original frame. The images from the top left in an anticlockwise direction show the incremental Percentage PPQ levels from 0-5. In the video the geese are flying from left to right. It can be observed that the geese are picked out for the 5x5 convolution filtering, the darker boxes, at PPQ0 level, including the straggler geese at the rear of the skein. The lower significance cloud edges are also being selected for 3x3 filtering, the lighter colour. As the PPQ level increases, more of the sharper edges of the cloud formations become more significant and selected for 5x5. Note that the filters were slightly amended to show the dark colour for the sobel 5x5 and the lighter shade for the 3x3 sharpening filter applications. In a real filtering case without this enhancement it is difficult to spot which area of the picture has been filtered, especially with video images.

5.5.3 *Demonstration Videos.*

In order to give more clarity to using the Demonstrator software, two youtube videos have been generated. The first video is around 20 minutes in length and shows the various stages of the processing, approximate significance, adaptive processing and application of DVFS. A link to the video can be found at:- [90] (Last checked 1 Oct 2019). The second video is around one minute 20 seconds and shows a short 20 second video without audio. The clip is then shown processed by the software demonstrator and finally the raw video is shown with audio enabled. The link to the video can be found at ref: [91] (Last checked 1 Oct 2019). The point of this is to demonstrate an issue pointed out in Chapter 1, the introduction to this thesis concerning CCTV operators and their difficulties having to scan recorded video for events, usually over a much longer time-frame and without recorded audio information. The adaptive processing and approximate significance highlights elements that can easily missed by human perception.

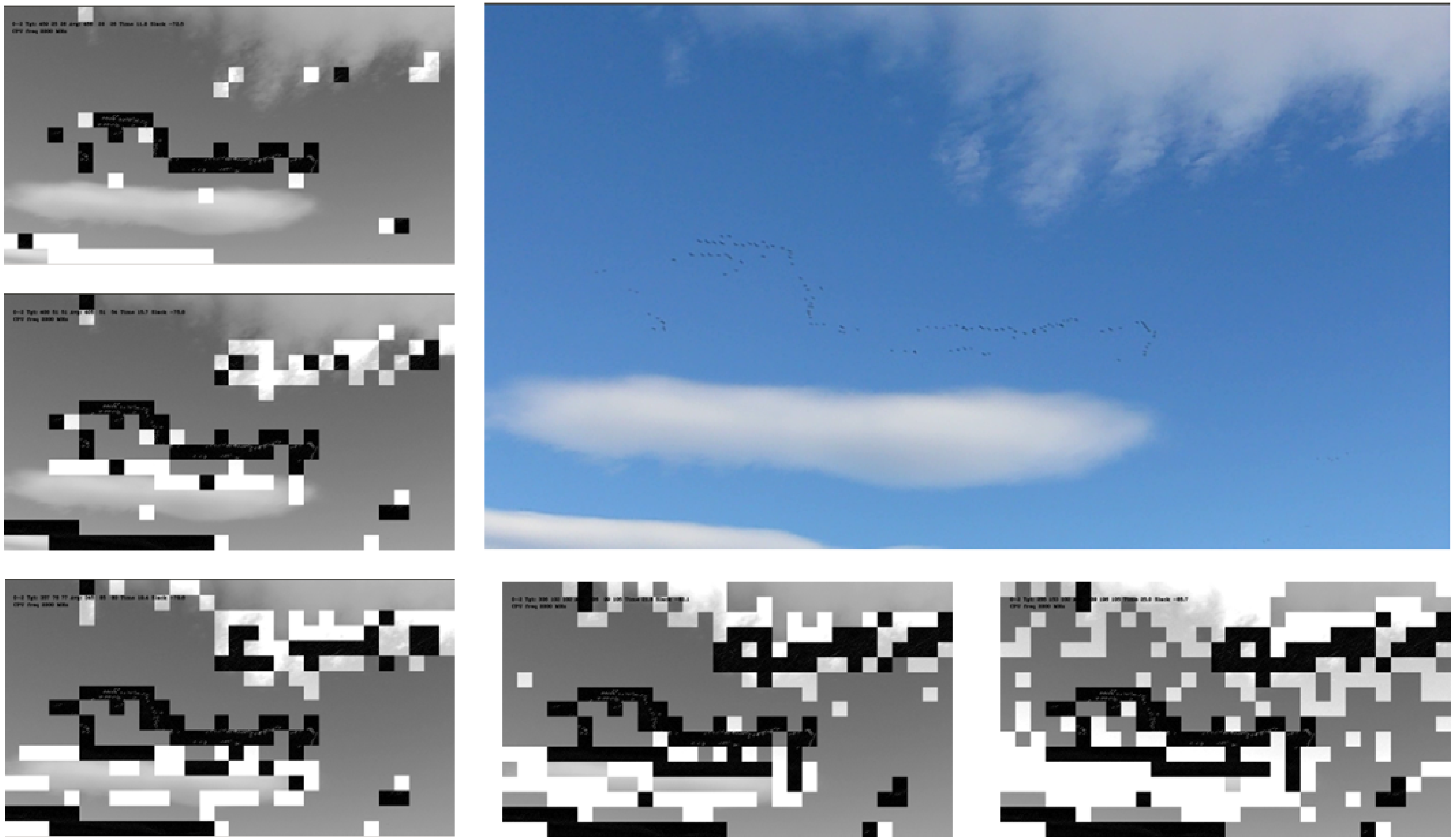


Figure 5.3: Selection of images from adaptive significance processing. Top right, in colour, is a frame from the Geese video. From top left in anticlockwise order is shown the processed images with PPQ0 90:5:5&, PPQ1 80:10:10%, PPQ2 70:15:15%, PPQ3 60:20:20% and PPQ4 50:30:20%.

5.5.4 *Application of DVFS*

The DVFS facility is only utilised on the Odroid XU4 and implemented purely for the exynos 5422 processor. This shows the current CPU frequency selection in conjunction with the CPU frequency slider in the sliders window. The actual Operating Performance Points (OPP) ie. combinations of frequency and core voltages are determined by the XU4 Linux device-tree as specified by the CPU manufacturer (Samsung). There are warnings from Linux community that separate voltage and frequency combinations are not to be experimented with by the user.

The DVFS facility allows experimentation to take place with a range of video files and live camera input, along with varying the PPQ levels, while monitoring the slack time displayed in the tell-back information. While the slack time is positive there is an opportunity to lower the CPU frequency via DVFS using the slider bar. Once the slack time turns to a negative value it can be observed that there will be some image stuttering due to a frame being missed, the frequency is then increased by one step until it becomes positive again and it can be noticed that the video image returns to a non-stuttering scenario and the optimal frequency for power reduction has been found.

5.5.5 *Evaluation of Performance and Power Consumption with DVFS*

At this point it is useful to quantify performance and power consumption at the various PPQ levels (6:0). The Cedar wax wing WQXGA image, 2560 x 1600 pixels, was used in the 10fps update mode. Table: 5.3 shows the execution and slack times for PPQ6 down to PPQ0 where it can be seen the 5x5 convolution is in negative slack time, for a 10 fps (100mS) update rate, with 150mS run time. Where there is a positive slack time there is opportunity for applying DVFS in order for the calculation period to just fit in the frame time. In order to explore the application of DVFS at the PPQ levels which have a positive slack time, PPQ6 is discounted for the negative slack time, then while reducing quality from PPQ5 down to PPQ0, initially record the power of PPQ5 at full CPU frequency of 2.0GHz for comparison. The Texas Instrument INA231EVM module was again utilised for power measurement as detailed in Chapter 4. The DVFS frequency was reduced at each of the PPQ (5:0) levels, to the point where the slack time stays just positive, around 10mS, and the power level was recorded. This sequence was followed by the same set of frequencies applied to running Linux only, in idle mode and the idle power recorded. These pairs of figures were then subtracted to evaluate the actual energy on image processing alone rather than Linux OS plus the application.

Processing and slack time for PPQ levels								
PPQ level		PPQ6	PPQ5	PPQ4	PPQ3	PPQ2	PPQ1	PPQ0
Processing time	mS	150.5	70.9	68.5	63.7	54.6	47.5	36.7
frame time	mS	100.0	100.0	100.0	100.0	100.0	100.0	100.0
Slack time	mS	-50.5	29.1	31.5	36.3	45.4	52.5	63.3

Table 5.3: Execution and slack times over range of PPQ(6:0).

On the initial trial and examination of the results the power level for all the PPQ levels, (6:0), at full 2.0GHz CPU frequency did not show any variation whereas at PPQs 5 down to 0 and implementing CPU frequency reduction on each, the power level reduced as was expected. It was then realised that, while the software was developed with threading in mind, that OpenCV 3.3 does not yet support threading or the chrono timing facilities. Therefore while waiting for for the next frame update or the one second timer the software is continuously running round an idle loop awaiting the next action. When DVFS is applied, since the processor is now running slower, the idle or slack time is reduced and proportionally more time is expended on the image processing. Only when the slack time approaches turning negative and potential image stuttering occurs does a truer figure for the power consumption emerge.

A further disruption to power measurement emerged while measuring the Linux Idle power with DVFS applied. The exynos is a "big.LITTLE" heterogeneous multi-processor system containing two clusters of four processors, "big" containing four A15 cores, CPUs 4-7, running from 200Mhz in 100MHz steps up to 2.0 GHz and "LITTLE" containing four A7 cores, CPUs 0-3, running similarly from 200MHz in 100MHz steps up to 1.4 GHz. DVFS is applied to all CPUs in a cluster but in the software demonstrator is only applied to the "big" cluster of A15 cores. Consequently as the CPU frequency is lowered, Linux shares the OS tasks around the other big cores and eventually onto the LITTLE cores as the big cluster frequency reduces sufficient for performance of Linux to be faster on the LITTLE cores. This situation will obviously have some effect on the Power figures but on average the same sort of event will happen in both the Linux Idle and Linux + App measurements and the overall difference figure should give a good representation of the App only power.

Table: 5.4 shows a snapshot of the power consumption across the range of seven CPU frequencies while running the demo software, with PPQ (5:0), on Linux plus the power while the Linux system was sitting Idle during DVFS. The Application Power is created by subtracting Linux-Idle Power from the Linux + App power in each frequency case. While the scenario of Linux moving OS modules onto other cores the App continues on the same core so the figures should give a reasonable representation of the actual energy savings.

Power level (Watts) for DVFS with PPQ									
PPQ level			PPQ6	PPQ5	PPQ4	PPQ3	PPQ2	PPQ1	PPQ0
DVFS		Unit	No	Yes	Yes	Yes	Yes	Yes	Yes
Frequency		GHz	2.0	1.5	1.4	1.2	1.1	0.9	0.7
Operation	App+Linux	Watts	9.15	6.00	5.71	5.30	5.23	4.79	4.58
	Linux-Idle	Watts	4.10	3.56	3.47	3.41	3.36	3.26	3.28
Difference	App Only	Watts	5.04	2.44	2.24	1.89	1.87	1.53	1.31

Table 5.4: Power consumption over DVFS range for PPQ(6:0).

5.6 technology roadblocks

5.6.1 *OpenCV, Mali SDK and Compute Library*

The previously mentioned Case study utilised the ARM Mali SDK to generate an OpenCL based solution on the Odroid XU4 platform to successfully demonstrate GPU computation of the 3x3 and 5x5 kernel filters on a small demo image. Shortly after completion of the Case Study, ARM withdrew the Mali SDK from public access, reserving it for their customers to use to produce a customised solution to developers in driving their GPUs etc. At the same time ARM launched the Compute Library, a customised version of OpenCV style functors with pre-compiled OpenCL modules, some of which are ARM equivalents to OpenCV functionality. However the compile tools, SCONS, and Paths for Compute Library are not compatible with the tools, CMake, used for OpenCV and mixing functionality of OpenCV and Compute Library is therefore not easily feasible. In addition Compute Library currently relies on the C++11 Chrono library and thread support, so due to time constraints it isn't possible to pursue a solution to this, ARM Community discussion forum (checked 20/Sep/2019) refers. In addition, while there are some limited examples which were able to be developed for the performance and energy testing in Chapter: 3, there is limited detailed documentation on the detail of using the library functions. It is anticipated that someone is probably working in the Open Community to solve these problems as the desire mixing the two together is an often asked question but always receives a negative response.

5.6.2 *Threads and Timer Interval Programming*

The active part of the demo would benefit from thread programming in order to process a new frame when it either becomes available, using camera, or when inter-frame timing requires fetching the next frame when using a video file, also when a one second time interval expires to update the Pool Thresholding when controlling it by the slider bars. Thread programming would also have been of benefit to performance for the image processing section for the paralleling of the convolution kernels in the demonstrator software. Threads and Chrono timer interval support has been introduced in C++11 but support by OpenCV is only implemented with version 4, which is only a beta release at the time of writing, full release is expected shortly.

5.6.3 *Utilisation of GPUs, DSPs, FPGA*

5.6.3.1 *GPU & DSP(Neon)*

The results shown in Chapter: 4, Section: 4.5.2 demonstrate that further significant process acceleration and opportunities for energy saving could be available if it had been possible to integrate OpenCV and Arm Compute Library and migrate the 3x3 and 5x5 convolutions to the Neon DSP or Mali GPU. This would have enabled further investigation of the application of DVFS to both the CPU system and the GPU in concert.

Energy usage					
	Total mJ/frame			Energy savings	
PPQ level	OpenCV	Neon	GPU	Neon vs OpenCV	GPU vs OpenCV
0	57.4	21.1	5.3	63.2%	90.8%
1	66.4	23.9	6.1	64.0%	90.7%
2	75.4	26.8	7.0	64.5%	90.7%
3	84.5	29.6	7.9	65.0%	90.7%
4	88.8	31.2	8.2	64.8%	90.7%
5	91.3	34.6	8.4	62.2%	90.8%
6	186.0	58.4	17.6	68.6%	90.5%

Table 5.5: Demonstration of the potential Energy savings over OpenCV if Neon or GPU operation could have been achieved.

As a theoretical 'what if' exercise a large spreadsheet calculation was created to explore what the energy and execution times, without DVFS, measured in Chapter: 4, Section: 4.5.2, would be for the Cedar Waxwing picture size image. The abbreviated results are shown in Tables: 5.5, 5.6 for Energy and compute time per frame. Obviously with such savings on compute time a larger amount of slack becomes available, permitting further energy savings with DVFS.

5.6.3.2 *FPGA*

FPGAs have demonstrated a track record of optimal performance and low energy usage as system accelerators and would offer another

Compute performance					
	Frame time mS			Compute savings	
PPQ level	OpenCV	Neon	GPU	Neon vs OpenCV	GPU vs OpenCV
0	24.7	10.7	2.3	57%	91%
1	27.5	11.5	2.5	58%	91%
2	30.3	12.3	2.7	60%	91%
3	33.2	13.1	2.9	61%	91%
4	34.2	13.5	3.0	60%	91%
5	31.8	14.4	3.1	55%	90%
6	68.5	21.1	5.2	69%	92%

Table 5.6: Demonstration of potential computational savings over OpenCV if Neon or GPU operation could have been achieved.

exploration area for this work. Previous personal experience of utilising FPGAs in both high volume, data streaming and processing operations, while significantly reducing energy consumption, enables a vision of executing approximate absolute significance while performing back to back reads concurrently. Betkaoui, [92] highlights the fact that FPGAs often operate at greater efficiency than GPUs in terms of performance per Watt. Kesturt, [93], also demonstrates the FPGA using an order of magnitude less power than a GPU for equivalent Linear Mathmatical processing operations.
The application of DVFS in FPGA has been a long standing target for research. Chow [94] demonstrates an experimental approach for a user to apply Dynamic Voltage Scaling (DVS), outside the specified operational limits, to effect lower power consumption on an FPGA. However Dahir [95], illustrates experimentation on the topic of full DVFS is still at a very early stage of exploration and development and as such is a non-trivial task.

5.7 SUMMARY

We have demonstrated a novel runtime methodology concept for energy efficient, Adaptive, Image Processing utilising Approximate Significance by utilising two percentage based thresholds, generating three levels of significance. Further energy efficiencies were also demonstrated by application of DVFS on the CPU to minimise the slack time.

This project was unable to proceed to a GPU or Neon implementation or even a multiprocessor implementation due to the roadblocks previously mentioned but as the OpenXXX tools develop, we are convinced that this demonstrator will be able to proceed to that stage in the near future.

Further efficiencies may be made available, when the libraries mature, to enable use of parallel programming with threads and enable use of available hardware to accelerate performance and reduce power by utilising GPU and Neon DSP.

The structure of the demonstrator program mimics the operation of ARM Compute Library, OpenCL based, examples, ie divides the work-area into 64x64 work-groups and submits a queue of tasks to the GPU. It is felt that this process may be adding some overhead in OpenCV when comparing it with the normal "whole image" filtering that is normally applied in OpenCV. If this is true, the timing results derived in Chapter 4 already include such overhead and would, therefore, provide a greater efficiency in performance and energy demand than that illustrated in this chapter. A further feature that may be introducing extra overhead is the requirement by OpenCV to have to pad out the edges of the image if the image dimensions are not integer divisible by the work-group dimensions. In this demonstrator we chose to add extra rows and columns at both sides of the image. We could have equally cropped the image down to the next divisible size but there is still some computing effort required. In the ARM Compute Library, no such action is required as the OpenCL/GPU interface handles this automatically.

5.8 FUTURE OPPORTUNITIES

The software model presented here has potential to be developed with further research in a number of ways in heterogeneous systems utilising a mixture of technologies, MultiProcessor, GPU, DSP, FPGA and perhaps ASIC.

5.8.1 *Multi Level Threshold Operation*

A multi-threshold significance level demonstrator extension could be implemented from the current level 0 to 2, to offer higher levels of image quality filtering by applying:-

- level 3, 7x7 kernel, or perhaps 3x3 followed by 5x5
- level 4, 9x9 kernel, or 5x5 followed by 5x5
- level 5, 11x11 kernel or 5x5 followed by 7x7

Another variant of this scheme would be to allow selection of the single higher order filter or the dual sequence filter, within the selected level. This scheme could provide a flexible approach to permitting computationally expensive higher order filtering to a minor proportion of highest significance areas in an image, while minimising energy usage.

A further exploration for energy saving in this area could include utilisation of Sage/Paraprox approximate kernels, Samadi [61, 44], in the convolution filter phase.

5.8.2 *Approximate Significance on Three Dimensional RGB Images*

In the early stages of the standard deviation Integral images research, it was decided to explore applying the standard deviation process to an RGB image as well as a gray sale image, a larger 20Mpixel image was being used to explore the computational time relative to smaller images. This resulted in a processing time of around 1 second for gray scale and 3 to 4 seconds for an RGB version of this image, so it was decided to stay with gray scale images for the research. Now that we have seen the impact of approximate deviation it would be worth re-exploring an RGB and/or YUV version of this process.

5.8.3 *Slack Time Driven DVFS and Energy Modulated Computing*

The demonstrator with the percentage levels controlling the processing could be adapted to automatically lower the DVFS frequency to keep the slack time just positive. A further development could be envisaged in an energy scavenging application where the controlling factor is the DVFS frequency being constrained to conserve energy and the percentage levels are varied to ensure the image is processed, keeping the slack time positive. If sufficient scavenged energy becomes available the DVFS constraint could be relaxed, thereby adaptively allowing higher percentage levels of data to be selected for processing.

5.8.4 *Arm Compute Library Version of Demonstrator*

A further step would be to implement the software principles used here in an ARM Compute Library version, a major rewrite of the code, in order to further explore the use of GPU and Neon frameworks. This would require a considerable learning period and may well throw down new challenges in the implementation phase. For example, the software examples provided by Compute Library illustrate sequentially well ordered kernel filters, being performed on a whole image such as 3x3, followed by 5x5. One remaining question is how will the memory manager cope when 1x1, performed by the CPU, 3x3 and 5x5 kernels,performed by the GPU, are thrown at targeted parts of the processed image contemporaneously?

5.8.5 *Energy and Performance Characterisation with CPU, GPU DVFS*

The energy and execution time characterization performed in chapter 4, was performed at the Maximum OPPs of both the CPU and GPU. The resultant experiments, data analysis/calculation times involved a fairly large Excel spreadsheet activity, consuming around two weeks of analysis time. The CPU is capable of 19 distinct OPP levels and the GPU has 10 OPP levels. In the case of being eventually able to run the demonstrator model utilising the various combinations of 9 image sizes with the 20 CPU OPPs and 9 GPU OPPs with both 3x3 and 5x5 kernel filters running on GPU and Neon, around 6500 combinations, it would be useful to characterise the CPU and GPU performance and energy requirements over this total range. However it can be appreciated that this would require some serious planning, benchmark testing, power measurements and experimental time. Ideally the whole activity would need to be, perhaps, performed in a Python based activity in order to automate the run time, data capture and analysis. This would eventually enable a fully Artificial Intelligence (AI) based prediction of OPPs to provide optimal performance or energy demand based on implementation requirements.

5.8.6 *Image Data Compression Applications*

Another potential opportunity could be to address the possibility, in data compression, where the significant areas are extracted from the background image, compressed and forwarded as smaller entities, or possibly as an augmentation in the next evolution of the H.264 (AVC) interframe or H.265 (HEVC) coding tree unit based compression techniques.

5.8.7 *Application in AI and ML*

Arm recently released their ArmNN, neural networks library, for exploration of Machine Learning (ML), but is dependent on use of their Compute Library.

This could open up two distinct application areas. The first being possible use in the Training phase of machine learning techniques. The second is possible use in the inference framework of Machine Learning eg category identification. While ArmNN is not currently available for embedded Cortex-M devices, that may change enabling instantiation in embedded applications at some future epoch.

5.8.8 *Large Scale Astronomical Image Applications*

Large data applications such as astronomical composite image processing where image sizes can be of immense proportions, for example the largest image released by Hubble of the M31 galaxy is a 4.3Gbyte file of a 1.5 Billion pixel image, which is evidently a cropped version of the full image. Adapting the existing software demonstrator to handle such large images would be a challenge and may require some other approach than, or enhancements for OpenCV.

5.8.9 *Image Database applications*

Could significance pattern combinations instead of Canny edges, be utilised to create a collection of tetrominoes [66, 67], become part of a signature file describing the content for image databases?

5.8.10 *Mobile Camera Applications*

Experimentation in this thesis has concentrated solely on a static camera position due to time constraints. There is an opportunity to explore use in a mobile camera applications, eg vehicle mounted.

5.8.11 *OpenMV Demonstration Application*

At the embedded level we have witnessed the arrival of OpenMV with the more recent arrival of the reasonably low cost H7 camera and an optional add on FLIR module. These elements operate under the microPython software environment and while they could viably be implemented in an embedded system, they may be a bit too power demanding for energy scavenging IoT instantiation, it can be envisaged that such a system may well be future-capable of operating under some form of OpenCV pre-compilation and ultimately with a GPU based system.

5.8.12 *OpenCV Library*

At some stage in the future the image significance extraction and adaptive processing functionality could be offered to OpenCV via the contributory modules where, depending on the demand of and usage by other users, they may be further optimised and offered as OpenCV modules in the main release.

6 CONCLUSION

6.1 SUMMARY AND CONCLUSIONS

6.1.1 *Background*

In Chapter: 1 We discussed the growing number of applications of IoT devices, increasing complexity and quantity of data produced, especially image data with migration to 4k and 8k. This will require an increase in backbone equipment and data centres to store and process large volumes of data. This scenario requires a trade off between "Fog" and "Cloud" computing in order to inteligently reduce data transmitted to the Cloud. Data compression provides a first line solution but creates extra burdens on the servers that have to decompress, process and re-compress image data. The escalating costs of servicing this demand is generating a requirement for a reduction in data rates, data storage and power. Thereby justifying energy efficient design, development and application of approximate computing techniques while maintaining an acceptable level of high performance. The research questions introduced in Section: 1.3 are restated here:

RQ1 Can we intelligently infer the significance of image blocks in Image Processing?

RQ2 Can we exploit the knowledge of significance to control computational complexity using application level approximation?

RQ3 Can we exploit a synergistic approach of hardware and software to maximise efficiency and reduce power consumption in heterogeneous systems in image processing workloads across different components?

6.1.2 *Solutions*

Addressing image processing in particular. Previous research has shown a myriad of solutions for determining significant areas of an image or video stream. Most of these techniques generally utilise some form of complex analysis to detect mobile features and significant areas by processing of a sequence of frames, utilising various convolution kernels, to detect and predict movement and isolate these features from static immutable background features.

The research techniques claimed in this thesis have created the following solutions to our Research Questions.

- Development of an image significance software technique that demonstrates ability to easily identify significance from a single frame, facilitating feature extraction. The technique can be utilised in full accuracy Standard Deviation mode, in a lower power Absolute deviation mode or in a further more energy efficient Approximate Absolute deviation mode.

- The Approximate Absolute significance extraction is combined with a novel approach to multi-level, significance based, image processing. A demonstrator software model, which we term Significance Driven Adaptive Approximate Computing (SDAAC), illustrates how convolution kernels offering more accurate 5x5 filtering can be applied to a selectable percentage of the most significant areas, the more normally used 3x3 filtering applied to a second selectable percentage level of a mid-range of significant areas and no filtering applied to the least significant areas, thereby creating a reduced computational demand with an accompanying reduction in energy demand. This distribution of the significance areas, offers a further opportunity with the potential of feature identification in Machine learning applications, in both learning and segmentation phases.

- The ensuing partial, multi-level, image filtering, renders potential computational and energy savings when compared with traditional whole frame image processing. The computational time saved can further be translated as slack time that can be utilised to control DVFS, permitting a maximised "just in time" energy efficient processing during the inter-frame period of video sequences. Further power measurement experiments utilising GPU and Neon FPU have illustrated how these heterogeneous elements can offer further potential for process acceleration with more efficient energy usage which could augment the SDAAC process outlined previously. Withdrawal of the Mali GPU development kit and Incompatibility issues between the OpenCV and ACL libraries prevented further development of the demonstrator to explore utilisation of such heterogeneous elements.

6.1.3 *Contributions*

The development of the significance techniques realised a previously undiscovered method which may have significant impact on future research and further development of existing techniques.
The application of standard, absolute or approximate absolute deviation with a local mean, presents a substantially economic method of

determining significance in an image.

This thesis has demonstrated a systems engineering approach to explore methodology that can aid reductions in computational and energy requirements for image processing by use of a collection of techniques to enable achievement of adjustable Power, Performance, Quality (PPQ) targets.

The concept of utilising multiple threshold significance levels to target a level of processing relevant to those significant areas of the image, present a new approach to processing and power efficiency. SDAAC presents a different approach to the traditional image processing techniques, offering a tunable multi level process as an alternative to traditional whole frame processing.

The use of the Odroid XU4 proved to be an extremely useful platform for the development of the software model, offering 4 off A15 and 4 off A7 CPUs along with associated Neon FPUs, a Mali T628 MP6 GPU and running a stable version of Ubuntu Linux along with the ability to apply DVFS to both the CPUs and GPUs. This in turn permits the opportunity for use of DVFS to gain further power savings.

6.1.4 *Findings*

These techniques, especially SDAAC, offer a number of new opportunities for areas of investigation as previously outlined in Section: 5.8, along with computational and energy efficiency.

The results presented in Chapter: 4 give concrete evidence of the processing and energy savings that can be achieved by the judicious use of GPUs and FPUs.

This year, 2019, has seen the release of stable versions of OpenCV version 4.x which now requires a C++11 compliant compiler and also allows optional C++11 compilation of Version 3.4.5 and above. Use of C++11 allows threading and <chrono> library timing event methods and the possibility of merging OpenCV with ARM Compute Library (ACL) in order to take advantage of heterogeneous hardware elements but still requires some further research to sort/merge the differences between CMake compilation for OpenCV and SConstruct for ACL.

6.1.5 *Improvement opportunities*

This work was centred around OpenCV3.3.0 libraries, it was not possible to achieve an integrated OpenCV, Arm Compute Library demonstrator to achieve GPU, Neon hardware and thread based solutions. Elements of this thesis have illustrated the energy saving and performance acceleration that could ultimately be achieved with the concept of Adaptive Approximate Significance with dedicated hardware and the use of parallel programming with threads. Further energy savings and performance enhancement could be achieved by deeper exploration in the use of FPGAs and DVFS. Other areas that would benefit the process is the use of Machine learning to present adaptive control of Power, Performance and Quality.

One of the personal motivational factors for starting this research concerns methodology to reduce data rates by having some form of intelligent camera that could identify relevant areas of an image and only transfer those delta segments with a full image frame being transferred on a less frequent period. This resultant research may well present the opportunity for further research, to reduce the image information transfer when utilising UHD cameras with embedded systems, especially in IoT applications.

6.1.6 *Self-Critique*

The project plan for this research anticipated risks involved with trying to utilise or merge the facilities offered with OpenCV, OpenCL and other library tools but unfortunately had not expected the nuances that would prevent the required merging of the facilities and achievement of a GPU/Neon based solution within the time-scales.

One issue that can arise among researchers is that, working at such an advanced level of their own topics that are possibly unrelated to each others'. In this case working with C++ with modern libraries, there may be no-one else working at the same level, meaning there are no peers to bounce ideas off or assist with reviewing code and there may be no available user guides or text books in print yet. In such events, topical websites and fora may have to be utilised to look for answers to specific questions but often have to be treated with caution! The end result may well be having to take a self-hypercritical approach along with careful experimentation to ensure any problems are addressed correctly and the solution is a valid answer to the problem. This, personally, is familiar ground as often, while working in an industrial capacity. Having to work outside a team to solve a critical challenge that others may have passed over due to the technical difficulty or lack of understanding of the problem! This often calls for a different approach that will yield an arguable solution that can be demonstrated and accepted by what may be typically a ′not so learned′ audience.

Part II

Thesis References

REFERENCES

[1] R. Wilson, "Rethinking the Internet of Things," *Syst. Des.*, apr 2015. [Online]. Available: https://systemdesign.intel.com/rethinking-the-internet-of-things/

[2] Unknown, "AI in IoT Market by Services & Software Solutions - 2024 | MarketsandMarkets," 2019. [Online]. Available: https://www.marketsandmarkets.com/Market-Reports/ai-in-iot-market-43388726.html

[3] T. K. Tan, R. Weerakkody, M. Mrak, N. Ramzan, V. Baroncini, J. R. Ohm, and G. J. Sullivan, "Video quality evaluation methodology and verification testing of HEVC compression performance," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 26, no. 1, pp. 76–90, 2016.

[4] G. A. Brady, N. Kapur, J. L. Summers, and H. M. Thompson, "A case study and critical assessment in calculating power usage effectiveness for a data centre," *Energy Convers. Manag.*, vol. 76, pp. 155–161, 2013. [Online]. Available: http://dx.doi.org/10.1016/j.enconman.2013.07.035

[5] A. H. Beitelmal and D. Fabris, "Servers and data centers energy performance metrics," *Energy Build.*, vol. 80, pp. 562–569, 2014. [Online]. Available: http://dx.doi.org/10.1016/j.enbuild.2014.04.036

[6] J. D. Ambrosia and S. G. Kipp, "2016 Ethernet Roadmap," 2016.

[7] Cisco, "Cisco GCI Highlights Tool http://tinyurl.com/y68psm93." [Online]. Available: https://www.cisco.com/c/en/us/solutions/service-provider/gci-highlights-tool/index.html

[8] N. C. Thompson and S. Spanuth, "The Decline of Computers as a General Purpose Technology : Why Deep Learning and the End of Moore ' s Law are Fragmenting Computing," *SSRN*, 2018. [Online]. Available: https://papers.ssrn.com/sol3/papers.cfm?abstract{_}id=3287769

[9] R. Wood, "The feasibility of magnetic recording at 1 Terabit per square inch," *IEEE Trans. Magn.*, vol. 36, no. 1, pp. 917–923, 2000.

[10] R. Wood, M. Williams, A. Kavcic, and J. Miles, "The feasibility of magnetic recording at 10 terabits per square inch on conventional media," *IEEE Trans. Magn.*, vol. 45, no. 2, pp. 917–921, 2009.

[11] R. W. Wood, J. Miles, and T. Olson, “Recording technologies for terabit per square inch systems,” *IEEE Trans. Magn.*, vol. 38, no. 4 I, pp. 1711–1718, 2002.

[12] A. Q. Wu, Y. Kubota, T. Klemmer, T. Rausch, C. Peng, Y. Peng, D. Karns, X. Zhu, Y. Ding, E. K. Chang, Y. Zhao, H. Zhou, K. Gao, J. U. Thiele, M. Seigler, G. Ju, and E. Gage, “HAMR areal density demonstration of 1+ tbpsi on spinstand,” *IEEE Trans. Magn.*, vol. 49, no. 2, pp. 779–782, 2013.

[13] Marko Kurt (Information Week), “Data Centre Decision Time,” *Inf. Week*, no. 1368, pp. 9–16, 2013.

[14] X. Wang, Y. Zhu, Y. Ha, M. Qiu, T. Huang, X. Si, and J. Wu, “An energy-efficient system on a programmable chip platform for cloud applications,” *J. Syst. Archit.*, vol. 0, pp. 1–16, 2016. [Online]. Available: http://dx.doi.org/10.1016/j.sysarc.2016.11.009

[15] S. Venkataramani, S. T. Chakradhar, K. Roy, and A. Raghunathan, “Computing approximately, and efficiently,” *Des. Autom. Test Eur. Conf. Exhib. (DATE), 2015*, pp. 748–751, 2015. [Online]. Available: http://ieeexplore.ieee.org/xpl/freeabs{_}all.jsp?arnumber=7092486

[16] M. Tatchell-Evans, N. Kapur, J. Summers, H. Thompson, and D. Oldham, “An experimental and theoretical investigation of the extent of bypass air within data centres employing aisle containment, and its impact on power consumption,” *Appl. Energy*, vol. 186, pp. 457–469, 2015. [Online]. Available: http://dx.doi.org/10.1016/j.apenergy.2016.03.076

[17] L. A. Barroso and U. Hölzle, “The Case for Energy-Proportional Computing,” *IEEE Comput.*, vol. 40, no. 12, pp. 33–37, 2007.

[18] D. Sidler, Z. István, and G. Alonso, “Low-latency TCP/IP stack for data center applications,” *FPL 2016 - 26th Int. Conf. Field-Programmable Log. Appl.*, 2016.

[19] P. Düben, J. Schlachter, P. ., S. Yenugula, J. Augustine, C. Enz, K. Palem, and T. N. Palmer, “Opportunities for Energy Efficient Computing: A Study of Inexact General Purpose Processors for High-Performance and Big-Data Applications,” *Des. Autom. Test Eur. Conf. Exhib. (DATE), 2015*, pp. 764–769, 2015. [Online]. Available: http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=7092489

[20] H. Hoffmann, S. Sidiroglou, M. C. S. Misailovic, A. Agarwal, and M. Rinard, “Dynamic Knobs for Responsive Power-Aware Computing,” *ASPLOS’11,*, 2011.

[21] B. H. C. Cheng, K. I. Eder, M. Gogolla, L. Grunske, M. Litoiu, H. A. Müller, P. Pelliccione, A. Perini, N. A. Qureshi, B. Rumpe, D. Schneider, F. Trollmann, and N. M. Villegas, "Using models at runtime to address assurance for self-adaptive systems," *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 8378 LNCS, pp. 101–136, 2014.

[22] M. Shafique, R. Hafiz, S. Rehman, W. El-Harouni, and J. Henkel, "Cross-layer approximate computing: from logic to architectures," *Proc. 53rd Annu. Des. Autom. Conf. - DAC '16*, pp. 1–6, 2016. [Online]. Available: http://dl.acm.org/citation.cfm?doid=2897937.2906199

[23] S. Mittal, "A Survey of Techniques for Architecting Processor Components Using Domain-Wall Memory," *ACM J. Emerg. Technol. Comput. Syst.*, vol. 13, no. 2, pp. 1–25, 2016. [Online]. Available: http://dl.acm.org/citation.cfm?doid=3014160.2994550

[24] V. Venkatachalam and M. Franz, "Power reduction techniques for microprocessor systems," *ACM Comput. Surv.*, vol. 37, no. 3, pp. 195–237, 2005. [Online]. Available: http://portal.acm.org/citation.cfm?doid=1108956.1108957

[25] G. Tagliavini, A. Marongiu, D. Rossi, and L. Benini, "Always-on motion detection with application-level error control on a near-threshold approximate computing platform," *2016 IEEE Int. Conf. Electron. Circuits Syst. ICECS 2016*, pp. 552–555, 2016.

[26] M. Rahimi, R. Baer, O. I. Iroezi, J. C. Garcia, J. Warrior, D. Estrin, and M. Srivastava, "Cyclops: In Situ Image Sensing and Interpretation in Wireless Sensor Networks," *Proc. 3rd Int. Conf. Embed. Networked Sens. Syst.*, pp. 192–204, 2005.

[27] I. Qiqieh, R. Shafik, G. Tarawneh, D. Sokolov, and A. Yakovlev, "Energy-Efficient Approximate Multiplier Design using Bit Significance-Driven Logic Compression," *Des. Autom. Test Eur.*, p. (in press), 2017.

[28] D. Esposito, A. G. Strollo, and M. Alioto, "Low-power approximate MAC unit," in *PRIME 2017 - 13th Conf. PhD Res. Microelectron. Electron. Proc.*, 2017.

[29] S. Venkataramani, A. Raghunathan, J. Liu, and M. Shoaib, "Scalable-effort classifiers for energy-efficient machine learning," *Proc. 52nd Annu. Des. Autom. Conf. - DAC '15*, pp. 1–6, 2015. [Online]. Available: http://doi.acm.org/10.1145/2744769.2744904{%}5Cnhttp://dl.acm.org/citation.cfm?doid=2744769.2744904http://dl.acm.org/citation.cfm?doid=2744769.2744904

[30] S. Venkataramani, S. T. Chakradhar, K. Roy, and A. Raghunathan, “Approximate computing and the quest for computing efficiency,” *Proc. 52nd Annu. Des. Autom. Conf. - DAC '15*, pp. 1–6, 2015. [Online]. Available: http://dl.acm.org/citation.cfm?id=2744769.2751163{%}5Cnhttp://dl.acm.org/citation.cfm?doid=2744769.2751163http://dl.acm.org/citation.cfm?doid=2744769.2751163

[31] V. K. Chippa, S. T. Chakradhar, K. Roy, and A. Raghunathan, “Analysis and characterization of inherent application resilience for approximate computing,” *Proc. 50th Annu. Des. Autom. Conf. - DAC '13*, no. i, p. 1, 2013. [Online]. Available: http://dl.acm.org/citation.cfm?doid=2463209.2488873

[32] V. K. Chippa, K. Roy, S. T. Chakradhar, and A. Raghunathan, “Managing the Quality vs. Efficiency Trade-off Using Dynamic Effort Scaling,” *ACM Trans. Embed. Comput. Syst.*, vol. 12, no. 2s, pp. 1–23, 2013. [Online]. Available: http://dl.acm.org/citation.cfm?doid=2465787.2465792

[33] V. K. Chippa, S. Venkataramani, S. T. Chakradhar, K. Roy, and A. Raghunathan, “Approximate computing: An integrated hardware approach,” in *Conf. Rec. - Asilomar Conf. Signals, Syst. Comput.*, 2013, pp. 111–117.

[34] V. K. Chippa, D. Mohapatra, K. Roy, S. T. Chakradhar, and A. Raghunathan, “Scalable effort hardware design,” *IEEE Trans. Very Large Scale Integr. Syst.*, vol. 22, no. 9, pp. 2004–2016, 2014.

[35] M. Van den Bergh, X. Boix, G. Roig, and L. Van Gool, “SEEDS: Superpixels Extracted Via Energy-Driven Sampling,” *Int. J. Comput. Vis.*, vol. 111, no. 3, pp. 298–314, 2015.

[36] D. S. Khudia, B. Zamirai, M. Samadi, and S. Mahlke, “Rumba An Online Quality Management System for Approximate Computing,” *Proc. 42nd Annu. Int. Symp. Comput. Archit. - ISCA '15*, pp. 554–566, 2015. [Online]. Available: http://doi.acm.org/10.1145/2749469.2750371{%}5Cnhttp://dl.acm.org/citation.cfm?doid=2749469.2750371

[37] A. K. Mishra, R. Barik, and S. Paul, “iACT: A Software-Hardware Framework for Understanding the Scope of Approximate Computing,” *Wacas*, 2014.

[38] Z. Wang and A. C. Bovik, “A Universal Image Quality Index,” *IEEE Trans. Signal Process. Lett.*, vol. 9, no. 4, pp. 81–84, 2002. [Online]. Available: http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=995823

[39] Z. Wang, A. C. Bovik, H. R. Sheikh, E. P. Simoncelli, H. R. S. Z. Wang, A. C. Bovik, E. P. Simoncelli, and E. P. Simoncelli, "Image quality assessment: From error visibility to structural similarity," *IEEE Trans. Image Process.*, vol. 13, no. 4, pp. 600–612, 2004.

[40] S. Hengstler, D. Prashanth, S. Fong, and H. Aghajan, "MeshEye: A Hybrid-Resolution Smart Camera Mote for Applications in Distributed Intelligent Surveillance," *2007 6th Int. Symp. Inf. Process. Sens. Networks*, pp. 360–369, 2007. [Online]. Available: http://ieeexplore.ieee.org/document/4379696/

[41] X. Hou and L. Zhang, "Saliency Detection: A Spectral Residual Approach," in *IEEE Conf. Comput. Vis. Pattern Recognit.* Minneapolis, MN, USA: IEEE, 2007.

[42] J. Ansel, Y. L. Wong, C. Chan, M. Olszewski, A. Edelman, and S. Amarasinghe, "Language and compiler support for autotuning variable-accuracy algorithms," *Proc. - Int. Symp. Code Gener. Optim. CGO 2011*, pp. 85–96, 2011.

[43] A. Raha, S. Venkataramani, V. Raghunathan, and A. Raghunathan, "Quality configurable reduce-and-rank for energy efficient approximate computing," *Des. Autom. Test Eur. Conf. Exhib. (DATE), 2015*, pp. 665–670, 2015. [Online]. Available: http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=7309615

[44] M. Samadi, D. A. Jamshidi, J. Lee, and S. Mahlke, "Paraprox: Pattern-Based Approximation for Data Parallel Applications," *Int. Conf. Archit. Support Program. Lang. Oper. Syst.*, pp. 35–50, 2014. [Online]. Available: http://dl.acm.org/citation.cfm?doid=2541940.2541948

[45] T. Simunic, L. Benini, and G. D. Micheli, "Energy-Efficient Design of Battery-Powered Embedded Systems," vol. 9, no. 1, pp. 15–28, 2001.

[46] I. Abdelkader, Y. El-Sonbaty, and M. El-Habrouk, "Openmv: a Python Powered, Extensible Machine Vision Camera," pp. 71–78, 2017. [Online]. Available: https://arxiv.org/ftp/arxiv/papers/1711/1711.10464.pdf

[47] H. T. K. Nguyen, H. Fahama, C. Belleudy, and T. V. Pham, "Low power architecture exploration for standalone fall detection system based on computer vision," *Proc. - UKSim-AMSS 8th Eur. Model. Symp. Comput. Model. Simulation, EMS 2014*, pp. 169–173, 2014.

[48] M. Rusci, D. Rossi, M. Lecca, M. Gottardi, E. Farella, and L. Benini, "An Event-Driven Ultra-Low-Power Smart Visual Sensor," *IEEE Sens. J.*, vol. 16, no. 13, pp. 5344–5353, 2016.

[49] J. F. Christmann, E. Beigné, C. Condemine, J. Willemin, and C. Piguet, "Energy harvesting and power management for autonomous sensor nodes," *Proc. 49th Annu. Des. Autom. Conf. - DAC '12*, p. 1049, 2012. [Online]. Available: http://dl.acm.org/citation.cfm?doid=2228360.2228550

[50] V. Raghunathan, A. Kansal, J. Hsu, E. Org, J. Friedman, and M. Srivastava, "Design Considerations for Solar Energy Harvesting Wireless Embedded Systems," *IEEE IPSN*, 2005. [Online]. Available: https://cloudfront.escholarship.org/dist/prd/content/qt76x92441/qt76x92441.pdf

[51] A. Rodriguez Arreola, D. Balsamo, A. K. Das, A. S. Weddell, D. Brunelli, B. M. Al-Hashimi, and G. V. Merrett, "Approaches to Transient Computing for Energy Harvesting Systems," *Proc. 3rd Int. Work. Energy Harvest. Energy Neutral Sens. Syst. - ENSsys '15*, pp. 3–8, 2015. [Online]. Available: http://dl.acm.org/citation.cfm?doid=2820645.2820652

[52] R. Shafik, A. Yakovlev, and S. Das, "Real-Power Computing," *IEEE Trans. Comput.*, vol. 67, no. 10, pp. 1445–1461, 2018.

[53] A. Yakovlev, "Enabling Survival Instincts in Electronic Systems: An Energy Perspective," *Transform. Reconfigurable Syst.*, no. May, pp. 237–263, 2015. [Online]. Available: http://www.worldscientific.com/doi/abs/10.1142/9781783266975{_}0013

[54] B. Grigorian and G. Reinman, "Dynamically adaptive and reliable approximate computing using light-weight error analysis," *Proc. 2014 NASA/ESA Conf. Adapt. Hardw. Syst. AHS 2014*, pp. 248–255, 2014.

[55] D. Mohapatra, G. Karakonstantis, and K. Roy, "Significance Driven Computation : A Voltage-Scalable , Variation-Aware , Quality-Tuning Motion Estimator," *Proc. 14th ACM/IEEE Int. Symp. Low power Electron. Des. - ISLPED '09*, p. 195, 2009. [Online]. Available: http://portal.acm.org/citation.cfm?doid=1594233.1594282

[56] F. Godtliebsen, J. S. Marron, and P. Chaudhuri, "Statistical significance of features in digital images," *Image Vis. Comput.*, vol. 22, no. 13, pp. 1093–1104, 2004.

[57] G. Karakonstantis, N. Bellas, C. Antonopoulos, G. Tziantzioulis, V. Gupta, and K. Roy, "Significance Driven Computation on Next-Generation Unreliable Platforms," *DAC*, pp. 290–291, 2011.

[58] B. K. Shreyamsha Kumar, "Multifocus and multispectral image fusion based on pixel significance using discrete cosine harmonic wavelet transform," *Signal, Image Video Process.*, vol. 7, no. 6, pp. 1125–1143, 2013.

[59] S. Mittal, "A Survey of Techniques for Approximate Computing," *ACM Comput. Surv.*, vol. 48, no. 4, pp. 1–33, 2016. [Online]. Available: http://dl.acm.org/citation.cfm?doid=2891449.2893356

[60] Q. Xu, T. Mytkowicz, and N. S. Kim, "Approximate Computing: A Survey," *IEEE Des. Test*, vol. 33, no. 1, pp. 8–22, 2016.

[61] M. Samadi, J. Lee, D. A. Jamshidi, A. Hormati, and S. Mahlke, "SAGE: Self-Tuning Approximation for Graphics Engines," *Micro*, 2013. [Online]. Available: http://cccp.eecs.umich.edu/papers/samadi-micro13.pdf

[62] Q. Zhang, T. Wang, Y. Tian, F. Yuan, and Q. Xu, "ApproxANN: An Approximate Computing Framework for Artificial Neural Network," *Des. Autom. Test Eur. Conf. Exhib. (DATE), 2015*, vol. 2015-, no. ii, pp. 701–706, 2015. [Online]. Available: http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=7092478

[63] X. Cui, Q. Liu, S. Zhang, F. Yang, and D. N. Metaxas, "Temporal Spectral Residual for fast salient motion detection," *Neurocomputing*, vol. 86, pp. 24–32, 2012. [Online]. Available: http://dx.doi.org/10.1016/j.neucom.2011.12.033

[64] R. K. Bania, "An effective supervised filter based feature selection algorithm using rough set theory," *2017 Int. Conf. Energy, Commun. Data Anal. Soft Comput. ICECDS 2017Bania2018*, pp. 2309–2314, 2018.

[65] A. Yakovlev, "Energy-modulated computing," *2011 Des. Autom. Test Eur.*, vol. 7, pp. 1–6, 2011. [Online]. Available: http://ieeexplore.ieee.org/document/5763216/

[66] G. Raghuwanshi and V. Tyagi, "Texture image retrieval using adaptive tetrolet transforms," *Digit. Signal Process. A Rev. J.*, vol. 48, pp. 50–57, 2016. [Online]. Available: http://dx.doi.org/10.1016/j.dsp.2015.09.003

[67] ——, "Feed-forward content based image retrieval using adaptive tetrolet transforms," *Multimed. Tools Appl.*, vol. 77, no. 18, pp. 23 389–23 410, 2018.

[68] R. Shafik, B. M. Al-Hashimi, and K. Chakrabarty, "Soft error-aware design optimization of low power and time-constrained embedded systems," *Des. Autom. & Test Eur. Conf. & Exhib. (DATE), 2010*, pp. 0–5, 2010.

[69] Intilop, "Full TCP & UDP Offload Engines with 1 - 16K Sessions Network Traffic and Application Acceleration in High Performance Network Servers and Storage Solutions," in *Ethernet Technol. summit*, no. April, 2015, pp. 1–19.

[70] J. P. Beckett, "Apparatus and method for digital camera and recorder having a high resolution color composite image output," 1998. [Online]. Available: https://patents.google.com/patent/WO1997046001A1/en?q=digital+camera+and+recorder{&}assignee=Beckett

[71] J. Han and M. Orshansky, "Approximate computing: An emerging paradigm for energy-efficient design," in *Test Symp. (ETS), 2013 18th IEEE Eur.* IEEE, 2013, pp. 1–6. [Online]. Available: http://ieeexplore.ieee.org/xpls/abs{%}7B{_}{%}7Dall.jsp?arnumber=6569370{%}0Ahttp://ieeexplore.ieee.org/xpls/abs{%}7B{%}25{%}7D7B{%}7B{_}{%}7D{%}7B{%}25{%}7D7Dall.jsp?arnumber=6569370

[72] V. Vassiliadis, K. Parasyris, C. Chalios, C. D. Antonopoulos, S. Lalis, N. Bellas, H. Vandierendonck, and D. S. Nikolopoulos, "A Programming Model and Runtime System for Significance-Aware Energy-Efficient Computing," *ACM SIGPLAN*, 2014. [Online]. Available: http://arxiv.org/abs/1412.5150

[73] B. K. Shreyamsha Kumar, "Image fusion based on pixel significance using cross bilateral filter," *Signal, Image Video Process.*, vol. 9, no. 5, pp. 1193–1204, 2015. [Online]. Available: http://dx.doi.org/10.1007/s11760-013-0556-9

[74] A. Kaehler and G. Bradski, *Learning OpenCV 3*, 2016.

[75] P. Viola and M. Jones, "Rapid object detection using a boosted cascade of simple features," *Comput. Vis. Pattern Recognit.*, vol. 1, no. C, pp. 511–518, 2001.

[76] ——, "Robust real-time face detection," *Int. J. Comput. Vis.*, vol. 57, no. 2, pp. 137–154, 2004. [Online]. Available: http://link.springer.com/article/10.1023/B:VISI.0000013087.49260.fb

[77] K. Preston, "The need for standards in image processing," *Nature*, vol. 333, no. 6174, pp. 611–612, 1988. [Online]. Available: http://www.nature.com/doifinder/10.1038/333611a0

[78] C. Leys, C. Ley, O. Klein, P. Bernard, and L. Licata, "Detecting outliers: Do not use standard deviation around the mean, use absolute deviation around the median," *J. Exp. Soc. Psychol.*, vol. 49, no. 4, pp. 764–766, 2013. [Online]. Available: http://dx.doi.org/10.1016/j.jesp.2013.03.013

[79] E. Totoni, M. Dikmen, and M. J. Garzarán, "Easy, fast, and energy-efficient object detection on heterogeneous on-chip architectures," *ACM Trans. Archit. Code Optim.*, vol. 10, no. 4, pp. 1–25, 2013. [Online]. Available: http://dl.acm.org/citation.cfm?doid=2541228.2555302

[80] D.-M. Bui, Y. Yoon, E.-N. Huh, S. Jun, and S. Lee, "Energy efficiency for cloud computing system based on predictive optimization," *J. Parallel Distrib. Comput.*, vol. 102, pp. 103–114, 2017. [Online]. Available: http://dx.doi.org/10.1016/j.jpdc.2016.11.011

[81] A. Aalsaud, R. Shafik, A. Rafiev, F. Xia, S. Yang, and A. Yakovlev, "Power-Aware Performance Adaptation of Concurrent Applications in Heterogeneous Many-Core Systems," *ISLPED*, vol. 16, 2016.

[82] W. F. Abaya, J. Basa, M. Sy, A. C. Abad, and E. P. Dadios, "Low Cost Smart Security Camera with Night Vision Capability Using Raspberry Pi and OpenCV," no. November, 2014.

[83] S. Nair, N. Somani, A. Grunau, E. Dean-Leon, and A. Knoll, "Image Processing Units on Ultra-low-cost Embedded Hardware: Algorithmic Optimizations for Real-time Performance," *J. Signal Process. Syst.*, pp. 1–17, 2017.

[84] O. S. Lawlor, "Embedding OpenCL in C ++ for Expressive GPU Programming," 2003.

[85] D. Connors, K. Dunn, and J. Wiencrot, "Adaptive OpenCL (ACL) Execution in GPU Architectures," 2013.

[86] S. A. Metwalli and Y. Hara-azumi, "SSA-AC : Static Significance Analysis for Approximate," *ACMTrans. Des. Autom. Electron.*, vol. 24, no. 3, 2019.

[87] L. Zhou, Q. Lv, and S. Guo, "A generic linux CPUFreq driver for ARM SoCs," *Int. J. Online Eng.*, vol. 9, no. SPECIALISSUE.6, pp. 29–32, 2013.

[88] D. Burke, "Methods for Energy Efficient Image Processing using significance learning, software demonstrator http://tinyurl.com/y5y6pavn," 2019. [Online]. Available: https://github.com/1dbup/ThesisSW

[89] Opengroup, "<unistd.h> getopt() http://tinyurl.com/y4zabvxr." [Online]. Available: http://pubs.opengroup.org/onlinepubs/7908799/xsh/getopt.html

[90] D. Burke, "How does Approximate Significance Image Processing work? http://tinyurl.com/yywjvpn2." [Online]. Available: https://youtu.be/kbKhU7CvEb8

[91] ——, "Application of Adaptive Approximate Image Significance http://tinyurl.com/y2f24j9z." [Online]. Available: https://youtu.be/rx6hIXdVcI4

[92] B. Betkaoui, D. B. Thomas, and W. Luk, "Comparing performance and energy efficiency of FPGAs and GPUs for high productivity computing," *Proc. - 2010 Int. Conf. Field-Programmable Technol. FPT'10*, pp. 94–101, 2010.

[93] S. Kesturt, J. D. Davis, and O. Williams, "BLAS comparison on FPGA, CPU and GPU," *Proc. - IEEE Annu. Symp. VLSI, ISVLSI 2010*, pp. 288–293, 2010.

[94] C. T. Chow, L. S. Tsui, P. H. Leong, W. Luk, and S. J. Wilton, "Dynamic voltage scaling for commercial FPGAs," *Proc. - 2005 IEEE Int. Conf. F. Program. Technol.*, vol. 2005, pp. 173–180, 2005.

[95] N. Dahir, P. Campos, G. Tempesti, M. Trefzer, and A. Tyrrell, "Characterisation of feasibility regions in FPGAS under adaptive DVFS," *25th Int. Conf. F. Program. Log. Appl. FPL 2015*, pp. 2–5, 2015.