Smart Camera Architectures for Wireless and Multi-Sensor Vision Applications

THÈSE N^{O} 9204 (2018)

PRÉSENTÉE LE 20 DÉCEMBRE 2018
À LA FACULTÉ DES SCIENCES ET TECHNIQUES DE L'INGÉNIEUR
LABORATOIRE DE SYSTÈMES MICROÉLECTRONIQUES
PROGRAMME DOCTORAL EN MICROSYSTÈMES ET MICROÉLECTRONIQUE

ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE

POUR L'OBTENTION DU GRADE DE DOCTEUR ÈS SCIENCES

PAR

Selman ERGÜNAY

acceptée sur proposition du jury:

Prof. A. Kis, président du jury
Prof. Y. Leblebici, directeur de thèse
Prof. R. Carmona-Galan, rapporteur
Prof. M. E. Yalcin, rapporteur
Prof. J.-Ph. Thiran, rapporteur

ÉCOLE POLYTECHNIQUE
FÉDÉRALE DE LAUSANNE

Suisse
2018

It is not a bug,
it is a feature.
— Anonymous

To my wife, Şerife…

Acknowledgements

First of all, I would like to express my sincere gratitude to my advisor Prof. Yusuf Leblebici for the opportunity he gave me to start this PhD work and his continuous support, patience, guidance and motivation. I learned a lot from him, not only from his broad research and engineering perspective, but also his personality. I am very thankful for the experience that I have acquired during the years of my PhD.

I am also grateful to the committee members, Prof. Andras Kis, Prof. Ricardo Carmona Galán, Prof. Müştak Erhan Yalçın and Prof. Jean-Philippe Thiran, for evaluating my work, their constructive comments and discussions.

I have found the opportunity of working with many people from different areas, and this allowed me to broaden my perspective. I would like to thank to the Master and Bachelor degree students with whom I have collaborated in different parts of my PhD work: Michaël Juillard, Alexis Michoud, Benoît d'Aramon, Benoit Alibert, Gökçen Nurlu, Jovan Blanusa, Hasret Sarıyer, Linda Özmen, Jeroen Buitendijk, Süha Köse, Yoann Biard, Çagri Erbağcı, Berk Olçum and Adrien Cellier.

I also would like to thank to my colleagues that I have collaborated with: Mustafa Kılıç, Kerim Türe, Bilal Demir, Jonathan Narinx, Vladan Popovic, Fırat Çelik, Ayça Akkaya, Duygu Kostak and Arda Uran. I also learned a lot from the discussions with Tuğba Demirci, Kerem Seyid, Ömer Çoğal and Kadir Akın and benefited from their experinces.

My thanks also go to the armasuisse team, Dr. Beat Ott and Dr. Peter Wellig, for their collaboration on the development and field test of the GigaEye-II system and on data collection for the polarimetry application.

Thanks to Dr. Alain Vachoux for his support on CAD tools and VHDL language, and Dr. Alexandre Schmid for his support with computing infrastructure. Thanks to Peter Brühlmeier and Sylvain Hauser for PCB design and mounting.

I would like to thank to my friends in LSM, Cosimo Aprile, Jonathan Narinx, Reza Ranjandish, Jury Sandrini, Behnoush Attarimashalkoubeh, Kiarash Gharibdoust, Elmira Shahrabi, Gain Kim, Sebastian Rodriguez, Wen-Yang Hsu, Igor Krawczuk, Clemens Nyffeler and Takao Marukame for funny and joyful moments we shared.

I am very much thankful to my close friend and officemate Mustafa and his wife Züleyha. With Mustafa, we shared a lot of fun, pleasant, enjoyable as much as difficult, stressful, tense moments during 5 years. His sincere friendship always helped me to enjoy any moment at the office. I also appreciate good friendships that I had with many people in our lab as well as at EPFL, namely, Ömer & Betül Çoğal, Kerem & Gizem Seyid, Kadir Akın, Şeniz & Deniz Eroğlu, Tuğba Demirci, Abdulhamid Kavranoğlu, Gülperi Özsema, Ayça Akkaya, Fırat Çelik, İrem Boybat, Duygu Kostak, Can Baltacı, Gözen Köklü & Ali Galip Bayrak, Işınsu Katırcıoğlu and Aysu Okur.

I have had the chance of having great people during my PhD life, to whom I am very much grateful for sharing memorable moments in Switzerland. First of all, I should acknowledge the unprecedented accompany and friendship of the couple Fatih & Filiz Karakoyun. Their unconditional presence made my PhD life more entertaining, pleasant and 'socialized'. I am also very much thankful to Ahmet & Elif Bekdemir, Enver & Berra Kılınç, Halil & Hilal Okur, Kerim & Edibe Türe for having unforgettable moments in our 'big' gatherings in Lausanne, which I always enjoyed very much.

Of course, I am very much grateful to my father Selahattin, my mother Naime, my sister Dilek and my brother Hakan. I always feel their unconditional support and love in every stage of my life.

And finally, I appreciate every moment I spent with my beloved wife, Şerife for standing with me during these challenging times, believing and motivating me. Without her support, it would be much more difficult.

Lausanne, 9 December 2018 S. E.

Abstract

Advances in camera sensor technology and its manufacturing process now allow high quality image acquisition with low-cost devices. Moreover, the latest significant increase in computational capacity of the processing units enables incorporation of more complex machine learning and deep learning methods within vision systems, expanding the capabilities of a typical camera system. A potential limitation of such complex and highly accurate machine learning and data processing methods is their high cost in terms of power and area. This limitation becomes more critical when multiple and/or wireless camera systems and come into question since such systems need to operate with limited power, memory and processing resources. Even though custom hardware solutions could solve this limitation problem, they however lack flexibility and hence are less practical. An embedded vision system with extended capabilities needs to be designed with a good trade-off between quality, speed, power consumption and flexibility.

A good trade off for an enhanced wireless multi-camera vision system may be provided by optimizing the system design at different levels. A common system-level approach to high-complexity systems is to partition the computational load and distribute it into local nodes. This corresponds to embedding computationally heavy operations into the camera units in a vision system which would reduce the bandwidth and overall power consumption. A camera equipped with a processing unit and memory that locally processes image data is called smart camera and can help overcome power, memory and processing resource limitations.

This thesis aims at designing a novel smart camera concept, and presents the hardware solutions to the proposed system design. Accordingly, in this thesis is proposed a flexible smart camera architecture which processes the pixel stream on-the-fly and produces metadata with low-latency, still providing high power and area efficiency. In particular, three processing blocks namely moving object detection, keypoint detection and description and cellular neural networks were implemented to illustrate the system design. In addition, proposed blocks are used in several applications such as omnidirectional image reconstruction, high resolution surveillance, polarimetry and wireless smart camera networks to show the flexibility of use of the proposed system in a wide-range applications.

Résumé

Les progrès sur la technologie des capteurs d'image et de son processus de fabrication permettent désormais l'acquisition d'images de haute qualité avec des appareils à faible coût. De plus, l'augmentation significative dernièrement de la capacité de calcul des unités de traitement permet l'incorporation de méthodes d'apprentissage automatique et d'apprentissage profond plus complexe au sein des systèmes de vision en étendant les capacités des systèmes d'imageur typiques. Le coût élévé en terme de taille et de puissance est une limitation potentielle de ces méthodes d'apprentissage automatique et d'apprentissage profond complexes et extrémement précis. Cette limitation devient plus critique lorsque plusieurs systèmes de caméra et/ou sans fil sont mis en question, puisque ces systèmes doivent fonctionner avec une puissance, une mémoire et des ressources de traitement limitées. Même si des solutions matérielles personnalisées pouvaient résoudre ce problème, elles manquent cependant de fléxibilités et sont donc moins pratiques. Un système de vision embarqué avec des capacités étendues doit être conçue avec un bon compromis entre qualité, vitesse, faible consommation de puissance et flexibilité.

Un bon compromis pour un système de vision multi-caméras sans fil amélioré peut être obtenu à travers l'optimisation de la conception du système à différents niveaux. Une approche haut-niveau commune sur les systèmes très complexes consiste à partitionner la charge de calcul et à la répartir dans des nœuds locaux. Ceci correspond à l'incorporation d'opérations lourdes de calcul au sein d'une seul unité de caméra dans un système de vision, ce qui réduirait la bande passante et la consommation énergétique globale. Une caméra équipée d'une unité de traitement et d'une mémoire qui traite localement les données d'image s'appelle caméra intelligente et peut aider à surmonter les limitations liés à l'alimentation, la mémoire et les ressources de traitement.

Cette thèse vise à concevoir un nouveau concept de caméra intelligente et présente les différenes solutions matériels pour la conception du système proposé. En conséquence, une architecture de caméra intelligente et flexible est proposé dans cette thèse, traitant le flux de pixels à la volée et produisant des métadonnées avec une faible latence, offrant toujours une grande efficacité en termes de puissance et de surface. En particulier, trois blocs de traitements nommément détection d'objets en mouvement, détection et description de points-clés, et réseaux de cellules neuronales ont été mis en place pour illustrer la conception du système. En outre, les blocs proposés sont utilisés dans plusieurs applications telles que la reconstruction d’image omnidirectionnelle, la surveillance haute résolution, la polarimétrie et des réseaux de caméras intelligentes sans fil pour montrer la flexibilité d’utilisation du système proposé dans une large gamme d’applications.

List of acronyms

ADC	Analog to Digital Converter
AHB	Advanced High Performance Bus
AMBA	Advanced Microcontroller Bus Architecture
APB	Advanced Peripheral Bus
ASIC	Application-Specific Integrated Circuit
BRAM	Block RAM
BRISK	Binary Robust Invariant Scalable Keypoints
CCD	Charge Coupled Device
CIF	Common Intermediate Format
CMOS	Complementary Metal-Oxide-Semiconductor
CNN	Cellular Neural Network
CNN-UM	Cellular Neural Network Universal Machine
CPU	Central Processing Unit
DAC	Digital to Analog Converter
DC	Direct Current
DDR	Double Data Rate
DoLP	Degree of Linear Polarization
FAST	Feature Accelerated Segment Test
FIFO	First In First Out
FOV	Field of View
FPGA	Field Programmable Gate Array
FREAK	Fast Retina Keypoint
fps	frame per second
FSM	Finite State Machine
FMC	FPGA Mezzanine Card
GUI	Graphical User Interface
GPU	Graphical Processing Unit
HD	High Definition

HoG	Histogram of Gradients
HPP	Hybrid Processor Population
IC	Integrated Circuit
I/O	Input/Output
IoT	Internet of Things
LBD	Local Binary Descriptor
LBP	Local Binary Pattern
LUT	Look-Up Table
LVDS	Low-Voltage Differential Signaling
MCU	Microcontroller Unit
MOG	Mixture of Gaussians
NMS	Non-Maximal Suppression
OpenCV	Open Source Computer Vision Library
ORB	Oriented FAST and Rotated BRIEF
PC	Personal Computer
PCB	Printed Circuit Board
PLL	Phase Locked Loop
PTZ	Pan Tilt Zoom
QDR	Quad Data Rate
RADAR	Radio Detection and Ranging
RAM	Random Access Memory
RF	Radio Frequency
RGB	Red-Green-Blue
RISC	Reduced Instruction Set Computer
SIFT	Scale-Invariant Feature Transform
SoC	System on Chip
SRAM	Static Random Access Memory
SURF	Speeded-Up Robust Features
UAV	Unmanned Aerial Vehicle
VGA	Video Graphics Array
VGA	Variable Gain Amplifier
VHSIC	Very High Speed Integrated Circuit
VHDL	VHSIC Hardware Description Language
WVSN	Wireless Visual Sensor Network
XGA	eXtended Graphics Array

Contents

Acknowledgements **v**

Abstract (English/Français) **vii**

List of acronyms **xii**

List of figures **xvi**

List of tables **xix**

1 Introduction **1**

2 State of the art **7**
- 2.1 Compound camera systems 7
- 2.2 Distributed camera systems 9

3 Architecture **13**
- 3.1 Overview 13
- 3.2 Moving Object Detection 15
  - 3.2.1 Background Subtraction 18
  - 3.2.2 Morphologic Filter 21
  - 3.2.3 Connected Components Labeling 23
  - 3.2.4 Result 28
- 3.3 Feature Extraction 29
  - 3.3.1 FAST Corner Detection 33
  - 3.3.2 FREAK Description 37
- 3.4 Hybrid Processor Population 42
  - 3.4.1 Cell Model 44
  - 3.4.2 Connectivity Rules 45
  - 3.4.3 Example Network Configurations 46
  - 3.4.4 Realization 48
  - 3.4.5 Conclusion 49
- 3.5 Result 50

4 Applications **53**

4.1 Omnidirectional Image Reconstruction . . . 53
4.1.1 Method . . . 53
4.1.2 Central Approach . . . 55
4.1.3 Distributed Approach . . . 57
4.1.4 Distributed and Parallel Implementation . . . 57
4.1.5 Interconnected Network of Cameras . . . 58
4.1.6 Novel hybrid architecture . . . 60
4.2 Polarimetry . . . 63
4.2.1 Method . . . 66
4.2.2 Result . . . 71
4.2.3 Conclusion . . . 72
4.3 Vision-based surveillance for drone detection . . . 74
4.3.1 System overview . . . 76
4.3.2 Result . . . 80
4.4 Wireless Smart Camera Networks . . . 82
4.4.1 Introduction . . . 82
4.4.2 Requirement Analysis . . . 83
4.4.3 System overview . . . 86
4.4.4 Electronics Design . . . 89
4.4.5 Conclusion . . . 92

5 Conclusion 93
5.1 Architecture . . . 93
5.2 Applications . . . 94
5.3 Future prospects . . . 94

Curriculum Vitae 109

List of Figures

1.1 (a) Illustration of camera obscura of Ibn al-Haytham (b) The first permanent photograph taken by Niepce . . . 1
1.2 Intelligence vs. bandwidth [4] . . . 4

2.1 (a) Stanford camera array [5] (b) Facebook surround 360 (c) Google street view [6] 8
2.2 Multi-camera systems with different dimensions developed in LSM/EPFL . . . 9
2.3 Smart camera examples . . . 10

3.1 Block diagram of the smart camera . . . 13
3.2 Image analysis chip of the smart camera . . . 14
3.3 Moving object detection pipeline . . . 17
3.4 Frame states for background subtraction block . . . 18
3.5 Advanced evaluation of the pixels . . . 20
3.6 Content of a memory row stores information of one pixel . . . 21
3.7 Example operation of a 2 × 2 erosion filter . . . 22
3.8 Hardware implementation of the erosion filter . . . 23
3.9 Data structure representing an instance . . . 24
3.10 FSM for connected component labeling . . . 25
3.11 Connected components labeling example operation . . . 27
3.12 Foreground detection results of the proposed method . . . 28
3.13 Feature extraction pipeline . . . 29
3.14 Difference of Gaussians in SIFT [60] . . . 30
3.15 Square box approximation in SURF [61] . . . 31
3.16 Comparison points in binary descriptor . . . 32
3.17 Hardware block diagram for local binary description . . . 33
3.18 A FAST corner on an example image . . . 34
3.19 Buffer used in FAST corner detection. Pixel numbers are for VGA resolution. . . 34
3.20 Hardware blocks for FAST corner test and score computation . . . 35
3.21 Buffering of scores for non-max suppression . . . 36
3.22 Hardware simulation results of the FAST block . . . 36
3.23 FREAK pattern applied on a corner . . . 37
3.24 FREAK pattern and box approximation . . . 37
3.25 Integral image calculation block . . . 38

3.26 Hardware of the box area computation 39
3.27 Hardware of the box area computation 40
3.28 Instantiating the box mean intensity computation block for FREAK pattern . . 41
3.29 Parallel comparison of mean intensities 42
3.30 FSM and timing diagram of metadata sequencer 43
3.31 Cell connections in CNN 43
3.32 A network configuration for LBDs 47
3.33 A network configuration for LBPs 48
3.34 Randomly distributed inhibitory and excitatory cells under limited connectivity rules. 48
3.35 Hardware of a single cell in HPP 49
3.36 Multiplexing the inputs for HPP 49
3.37 Hardware simulation environment of smart camera 50

4.1 (a)Cameras contributing to the direction $\vec{\omega}$, (b) contributing pixel positions on the image frame of the contributing cameras for direction $\vec{\omega}$, (c) projections of camera centers contributing to direction $\vec{\omega}$ onto planar surface normal to $\vec{\omega}$. . . 54
4.2 Centralized implementation approach to the reconstruction algorithm. Camera data is saved onto one of the SRAMs, via camera interface and memory interface. Panorama reconstruction algorithm access the camera data which belongs to previous frame. 56
4.3 High level model of an interconnected network of cameras. All cameras C_i are connected via interconnection network and some cameras have direct access to central unit. 58
4.4 (a) The network-based Panoptic prototype with 5 floors and 49 cameras. The sphere diameter of the prototype is $2r_{\odot} = 30$cm. (b) Top view of the Panoptic Media FPGA-based development platform. 59
4.5 Network with tree topology of depth 2. PN1 operates on images collected by C and PN2 stitch partially reconstructed images by PN1. Maximum supported number of cameras is determined by fanout of PN1 and PN2 (m, n, respectively) 60
4.6 (a) Front view of the horizontal board carrying FPGA, memories, power units and misc. devices (b) Back view of the horizontal board carrying 16 camera connectors 61
4.7 Hardware solution for a system which has single PN2. Cameras is placed on hemisphere, and collected images are transmitted to vertical boards which represents PN1. Omnidirectional image is completed on horizontal board, PN2, processing of stitched image by PN1. 62
4.8 Horizontally placed wire-grid polarizer absorbs the horizontal components. . . 63
4.9 Polarization camera examples 64
4.10 Super-pixel structure 65
4.11 Steps followed in this study 67
4.12 Raw polarimetric images 68
4.13 Transformed polarimetric images 69

4.14 Initial detection 70
4.15 Training loss 71
4.16 Detected objects 73
4.17 Drone detection methods 74
4.18 GigaEye-II, captured in drone detection days of armasuisse, 2016 77
4.19 GigaEye-II connectivity with single PC via 10 Gb ethernet 78
4.20 GigaEye-II connectivity with dual PC via PCIe 78
4.21 Smart camera concept applied to the GigaEye system 79
4.22 Operation of the drone detection software for GigaEye platform 79
4.23 Moving Object Detection stages 80
4.24 Moving Object Detection stages 81
4.25 Wide area surveillance with distributed camera nodes 82
4.26 Pin-hole camera model 83
4.27 WVSN node block diagram 87
4.28 FSM for operation of the WVSN 88
4.29 Block diagram of the processing chip 89
4.30 NAND Flash interface board equipped with FMC connector 89
4.31 Block diagram of ADC interface 90
4.32 Tapeout in 65-nm TSMC technology 91
4.33 3D view of the energy autonomous camera system 92

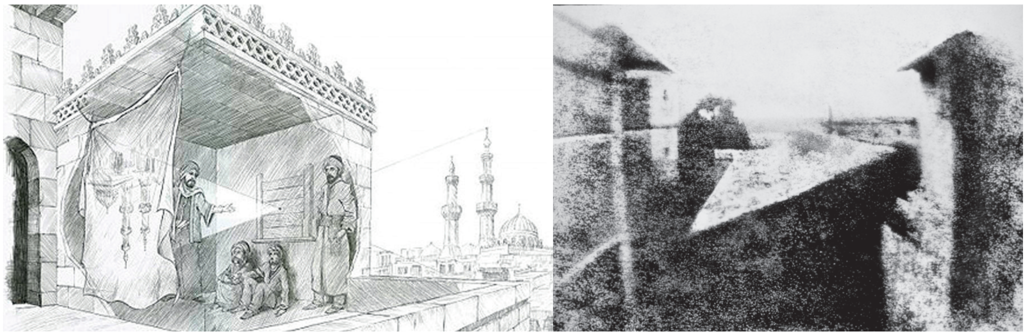
List of Tables

3.1 LUT approximation of $A \div r^2$ for FREAK pattern 39
3.2 Resource allocation 51
3.3 Performance comparison for Moving Object Detection 51
3.4 Performance comparison for Keypoint Description 51

4.1 Storage and bandwidth comparison of the Panoptic systems 62
4.2 Details regarding to the camera systems used in this study 65
4.3 Accuracy of the trained models 72
4.4 Wireless power analysis 85
4.5 Solar irradiance [kWh/m^2/day] on a horizontal surface in Lausanne [116] . . . 85
4.6 Harvested energy per day in the worst case 85

1 Introduction

Although development of the camera technology is presumed to belong to the last centuries, its roots go back to the fifth century BC in the ancient Greece. Micius, Aristotle and Euclid built the basic working principle of the pinhole camera. They found that the light travels as straight lines and when it passes through a pinhole, it projects an inverted image of the scene. In addition to this, they also realized that smaller pinhole aperture enables sharper projection [1]. Based on this principle, the first camera obscura was designed by Arab scientist Ibn al-Haytham in 9th century, illustrated in Figure 1.1a. This camera obscura, or pin-hole camera, forms the basis of photography. During the following centuries, many scientists have contributed to the development of the pinhole camera. The earliest cameras that are built until 17th century were room-sized and the produced images were only preserved by manual tracing. In the 17th century, first portable model is invented by Robert Boyle and Robert Hooke and the first permanent photographic image (see Figure 1.1b) was made by a French inventor Nicephore Niepce in the early 19th century [2]. The inventions regarding photographic process continued with Niepce's partner Daguerre who proposed a first practical photographic process, called Daguerrotype process which is followed by dry plates, photographic films etc.



(a) (b)

Figure 1.1 – (a) Illustration of camera obscura of Ibn al-Haytham (b) The first permanent photograph taken by Niepce

With the emergence of digital cameras, the technology has been progressed drastically. Today's digital camera era, two types of camera sensor are available in the market: Charge Coupled Device (CCD) and Complementary Metal-Oxide-Semiconductor (CMOS) sensors. CCD sensors which provides high quality and low-noise images require a special manufacturing process to transport the charge across the chip without distortion. Charge to voltage conversion is performed before the output buffer of the readout circuit. For this process, very high quality sensors with high light sensitivity is needed. After exposure is completed, CCD transfers charge packet of the each pixel sequentially via a shift register structure.

In the CMOS sensor technology, each pixel has several transistor next to it. This reduces the light sensitivity and consequently the image quality. However, CMOS sensors consumes much less power than CCD and its digital memory style readout faster readout circuitry allows to reach high frame rates. Moreover, its low-cost manufacturing process allows to produce much cheaper sensors and this makes the CMOS sensors widely spread.

Developments in memory, smart phone and cloud technologies together, make the storage of the photos or sharing them much easier. Moreover, advanced computer vision methods, image understanding capabilities of the computers with the help of graphical processing units are extremely increased and computationally heavy deep learning blocks can be operated on these powerful devices. Such advances in hardware and software units have brought new functionalities to the camera technology, making cameras more accessible to end-users and enlarging its use areas.

Even though the camera systems have evolved much more than expected in the last decades, there are still many open issues and challenges. One of the limitations of the today's camera systems is their limited angle of view. Although the available fish-eye lenses extend the angle of view, they inevitably bring distortion and non-uniform resolution. Curved sensor and micro-lens arrays could be an alternative solution but they require special manufacturing process making them cost-inefficient. Multiple camera systems can provide less-distorted and low-cost solution to angle-of-view limitation of the current camera systems.

Another issue is power efficiency of the camera systems. Wireless camera systems have been popular in the last decade due to its large number of application areas including surveillance. Wireless nodes in such systems are designed either as battery-operated or as self-powered. In either case, they have limited power budget. Therefore, this limitation usually does not allow broadcasting. As data to be transferred through such systems is limited, it should be preprocessed in advance and only salient information should be transmitted to the central units. This requires a wireless node to be able to process the frames locally before transmission.

Increasing number of cameras connected to a single processing unit may also suffer from bandwidth limitations. In wired systems, pin count of the processing unit, PCB path and connector types determines the bandwidth. In wireless systems, carrier frequency, transmission power and range are major delimiters.

Considering these limitations and issues, distributing the computational load into local nodes, or in other words, increasing intelligence of the nodes, gives a solution. In this case, each peripheral unit in a multiple and/or wireless camera systems has to be designed as data-aware cameras. Each camera is accordingly has its own processing unit operating according to the assigned task. The good organization of those cameras is able to eliminate the limitations of typical multiple and wireless camera systems. We refer such a data-aware camera with extended functionalities as *Smart Camera* and is the main focus of this thesis.

There are different definitions of smart camera used in the literature and industry [3]. For instance, auto-exposure or auto-focus can also be referred as smart camera. However, in this work we use the smart camera concept as the camera not only taking images, but also providing information about the image content by local processing. The objective of a smart camera system is not broadcasting or high quality imaging, therefore it performs in highly energy-efficient way and requires low bandwidth. A big advantage of a smart camera is obviously its appropriateness for wireless camera systems requiring low-bandwidth and low energy consumption. With the current trend of wireless communications, the necessity of designing energy and data-efficient wireless systems is high and smart cameras can provide good solution for such important camera systems.

The relation between the level of intelligence and bandwidth requirement is depicted in Figure 1.2. In a standard camera where only raw data is streamed, the bandwidth requirement is at its maximum since it lacks of the ability to local data processing, hence all the pixels captured are sent to the central processing unit. As the level of the intelligence increases by different data processing steps such as object detection, object description and event description, bandwidth requirement reduces. In those cases, the data to be transferred can be pixel information of the related area of the figures, keypoints of the area of interest or classification of the object of interest instead of the whole image scene. This could decrease the transmission bandwidth and processing load in the central units providing faster and more efficient data communication.

With the aforementioned capabilities, smart cameras have many application areas and new applications emerge with the development of electronics and software technology. For instance, with the emergence of drone technology, the use of smart cameras in drones is a new topic to be explored nowadays. Many similar applications will be appearing in the future, which makes the efficient design and use of smart cameras crucial for the literature and the industry. Applications of the smart cameras cover a wide range listed below, but not limited to:

- Surveillance for security
- Disaster-related surveillance
- Wireless healthcare applications
- Industrial monitoring and control

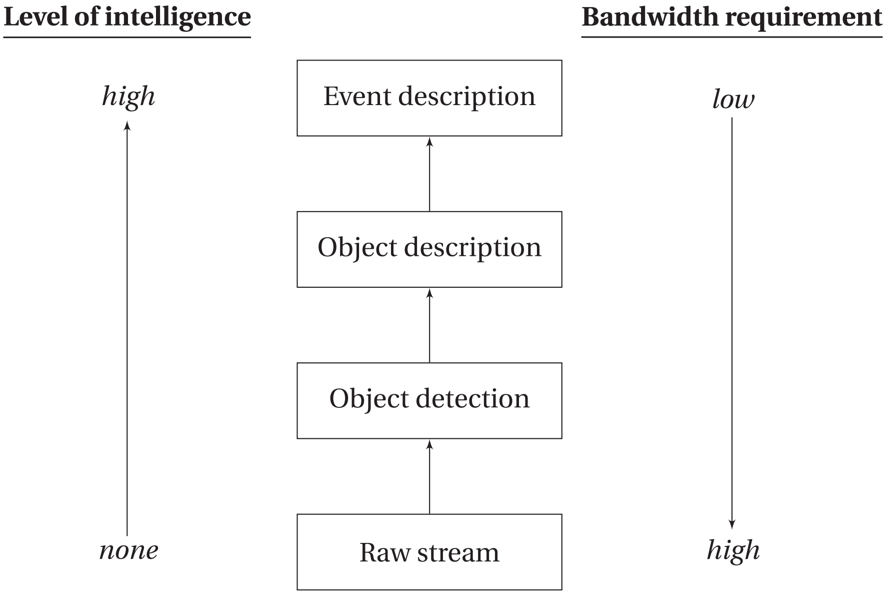


Figure 1.2 – Intelligence vs. bandwidth [4]

- Environmental and habitat monitoring
- Drone applications
- Simultaneous localization and mapping (SLAM)

In this thesis, we propose a smart camera architecture for wireless and multi-camera applications. This architecture aims at utilizing an external image sensor and performs its operations by processing the sensor pixel stream on-the-fly. Smart camera provides the output information as metadata with minimum latency. As processing blocks, we implemented moving object detection, keypoint detection, binary description, and Cellular Neural Network blocks. The smart camera system that we propose in this work is not the unique or exact solution to the problem. Yet, it provides a better tradeoff than its counterparts in terms of flexibility, power and timing efficiency.

Key contribution of this thesis

The details of the contributions regarding to the architecture and applications are below:

- **Architecture:**
  - A power and area efficient low-complexity moving object detection method is proposed and implemented in FPGA. Its algorithm which consists of background subtraction, morphological filtering and connected component labeling is detailed.

  - A resource efficient hardware implementation of a keypoint detection combined with local binary description is presented. Particularly, FAST corner detection and FREAK description algorithms are designed with VHDL and implemented for both FPGA and ASIC.
  - A CNN-like network, Hybrid Processor Population is integrated to the smart camera system. Different network configurations which can help local computations is given.
- **Applications:**
  - A hybrid architecture for multi-camera systems are proposed and its hardware is realized. Its PCB design consisting of FPGA and QDR-II memories are presented.
  - A high-resolution vision based drone detection method and its realization is presented.
  - A deep learning method is applied for polarimetry applications. We collected data from polarimetric cameras, train the networks by testing different polarimetry components. Finally we proposed a multi-camera polarimetry hardware.
  - A wireless smart camera node is proposed exploiting the smart camera blocks. System consists of Leon-3 open source RISC processor for system management, NAND Flash interface for data storage, and analog blocks for sub-1 GHZ wireless transmission.

Thesis outline

The outline of this thesis is as follows: in Chapter 2, we give a brief state of the art concerning the smart camera systems. Chapter 3, explains the components constitutes of our smart camera architecture. Particularly, moving object detection, keypoint detection and description and Cellular Neural Network blocks are explained and their hardware implementation details are given. In Chapter 4, 4 different example applications are given. Drone detection, omnidirectional reconstruction, polarimetric imaging and wireless smart camera network problems are examined and smart camera solutions are presented. Finally, Chapter 5 concludes the thesis and present future research directions.

2 State of the art

In Chapter 1, a brief introduction of the smart camera systems is given as well as our motivation to this work and the challenges is explained. In this chapter, state of the art in multiple and wireless camera systems will be presented. State of the art regarding the methods and algorithms used in this work can be found in related sections.

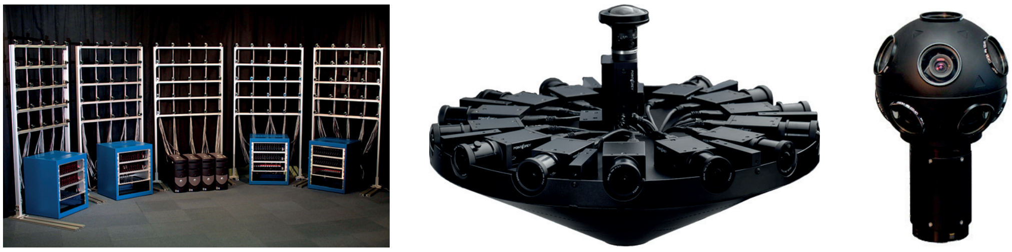
Considering the difference in application areas and challenges, multiple camera systems can be examined in two groups: (i) compound and (ii) distributed camera systems. While the compound camera systems usually target to obtain wide FOV images, in distributed camera systems environment is observed partially by each wireless node in a power efficient way.

2.1 Compound camera systems

One of the main restrictions of the single conventional cameras is their limited angle of view. Several solutions for acquiring omnidirectional images and their application have been presented in [7]. In fact, successful examples of wide angle viewing are available in the nature as well. For instance, compound eyes of common fly is an appropriate example, which contains hundreds of optical units providing panoramic field of view [8]. Ideally this structure can be mimicked by curved sensors, however since their manufacturing process is very difficult, this solution is very expensive and has low resolution [9].

Another way is combination of special lenses and convex mirrors, but distortion and non-uniform resolution is inevitable in this case [10]. Among the alternatives, multiple camera systems have been gaining importance in the recent years due to their benefits such as uniform and high resolution output, in addition to the reduced distortion. Moreover, the conventional cameras are broadly available and low-cost technology, therefore can provide much more cost-effective solution compared to the aforementioned methods requiring special manufacturing process.

Compound camera systems are composed of outward looking multiple cameras placed on a planar, spherical or cylindrical surface. A number of multiple camera systems are designed



(a) (b) (c)

Figure 2.1 – (a) Stanford camera array [5] (b) Facebook surround 360 (c) Google street view [6]

with different features in terms of dimensions, real-time capability, number of cameras, output format and resolution.

One of the first examples of compound camera systems is FlyCam [11] which offers a PC based stitching. It consists of 5 inexpensive and low-resolution cameras. The Stanford Camera Array [5] is the other early example of multiple camera systems. It consists of 100 cameras connected to 4 PCs as shown in Figure 2.1a. It has a limited processing capability in the camera level. The system is used for recording the videos to be later stitched offline.

Google Street View [6] is a popular example for polydioptric data acquisitions systems. 15 of 5-megapixel CMOS image sensors are placed on a spherical surface as shown in Figure 2.1c. It is a 360° imaging system comprising 15 5MP cameras, which covers 80% of its surroundings. This system is used to generate contents for street view property of Google Maps.

Another panorama system with high resolution output, OmniCam is presented in [12]. This scalable system supports up to 12 HD cameras, and output panorama is stitched in post-production.

In order to reach higher output resolution, number of cameras and sensor resolution can be increased. An example camera system which is able to acquire an image frame with more than 1 Gigapixel resolution was presented in [13]. The system uses a very complex lens system comprising a parallel array of micro cameras to acquire the image. Due to the extremely high resolution of the image, it suffers from a very low frame rate of three frames per minute.

Generating the panoramas offline is useful in many cases, but real-time stitching have been a demanding task especially in interactive applications. In [14] real-time systems with six cubically arranged cameras are presented. These systems utilize high resolution imagers with a low number of cameras.

The Panoptic Camera, introduced in [15] is capable of real-time data processing and omnidirectional view generation. Hardware implementation of different blending algorithms on this system is presented in [16]. The details of the Panoptic Camera is introduced in Section 4.1. Based on the proposed algorithm, different systems have been developed [16], [17], [18] and integrated with VR interface [19] in LSM (Microelectronic Systems Laboratory) at EPFL. Hard-

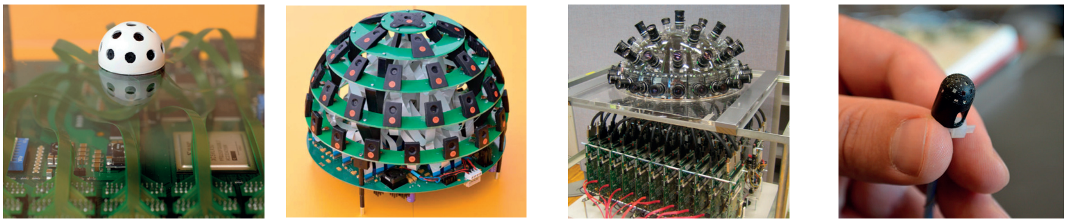

(a) Ping-pong (b) P49 (c) GigaEye (d) Endopano

Figure 2.2 – Multi-camera systems with different dimensions developed in LSM/EPFL

ware implementation of the system which has 15 cameras on a 3-layer and 3 cm diameter hemisphere can be seen in Figure 2.2a. In [18], high-resolution-44-camera system design for defense and security applications is presented. It is capable of recording omni-directional video in a 360° × 100° FOV at 9.5 fps with a resolution over 82.3MP (see Figure 2.2c). Another approach based on interconnected network of the cameras is proposed in [17], and the presented hardware is depicted in 2.2b.

Physical dimensions of the multiple camera systems can be reduced down to millimeter scale. Cogal and Leblebici developed a miniaturized omnidirectional camera system with 24 pinhole cameras [20], shown in Figure 2.2d. Cameras are placed on a 5 mm radius hemispherical case. Its FPGA implementation achieves 1 megapixel resolution at 25 fps.

2.2 Distributed camera systems

One of the earliest examples of the smart camera system is CMUcam [21] which was proposed in 2001. It combines a low-cost CMOS image sensor at CIF resolution and a low-cost microcontroller. Since the processing capability is quite limited, it can only perform a basic color based blob tracking at 16.7 fps.

Another example of early smart cameras with MCU is Cyclops [22]. It consists of a CPLD for image capturing from a CIF-resolution camera and transfer it to the frame buffer in 64-KB external SRAM. Since the image resolution $352 \times 288 = 99K$ is greater then the available memory, it reduces down the resolution to 128×128 for the processing applications. Buffered frame is accessed by a 8-bit microcontroller to perform some vision algorithms, including moving object detection and hand posture recognition.

The Cyclops and CMUCam is used as the low-power tier of SensEye [23] system which employs multiple tier of cameras. For the high-level tiers it has a VGA resolution web-cam and PTZ (pan-tilt-zoom) camera at HD resolution with high-cost. The network activates the high-level tiers, if only low-level tiers detect an important activity.

The processing capability of WVSN nodes is increased with the development of more efficient MCUs. 32-bit ARM powered MeshEye [4] system is battery-operated and provides a low-

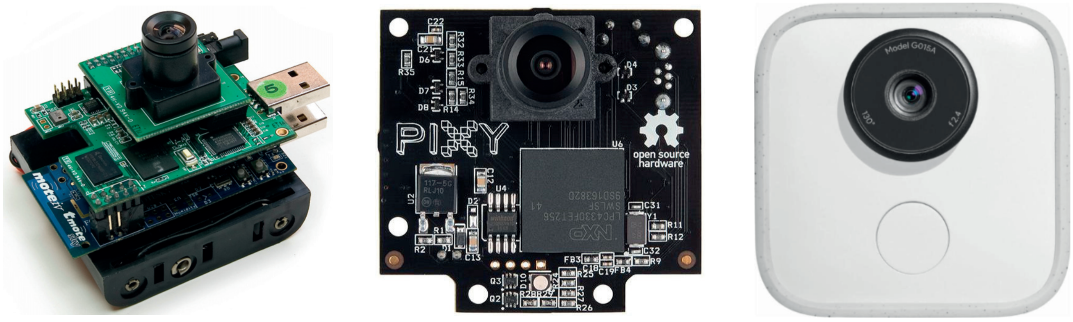

(a) CITRIC (b) CMUcam-5 (c) Google clips

Figure 2.3 – Smart camera examples

resolution stereo vision, presented in 2007. In this system, a low-resolution stereo camera system observes the scene and determines the position, range and size of the moving objects. This information triggers a high resolution color camera module for further analysis.

Increasing CPU power and memory extend the capability, thus application range. Integrating an Intel XScale MCU which can operate up to 624 MHz, 16 MB Flash and 64 MB RAM in CITRIC [24] system enables image compression, target tracking and camera localization applications. Its power consumption is reported to be minimum 428mW at IDLE mode, and 970mW at high performance mode.

Combining the high performance MCUs and open source software tools provides a high flexibility in terms of the applications. Raspberry Pi (ARM 700 MHz) and MSP430 MCUs are used together in a solar powered wireless smart camera SWEETcam [25], and proposed for the surveillance of public spaces in 2014. It also benefits from the Linux operating system and OpenCV libraries which reduce the application development time significantly. But the power consumption is still around 500 mW.

In the recent years, commercial products targeting consumer electronics market have been appearing. A popular example is Google Clips [26] shown in Figure 2.3. It selects the best moments with a learning algorithm and store them as short video files. However, its battery enables up to only 3 hours operation.

Energy consumption could be reduced by custom hardware design at the expense of software flexibility. FPGAs are reconfigurable digital devices which allows parallelization of task. Thus, lower operating clock frequencies reduce power consumption. An early smart camera architecture with FPGA is presented in 2006 [27]. Inspiring from the human vision system, authors split the process into three layers: (1) attention and (2) focusing of eye sends the pertinent information, and it is (3) interpreted in brain. Since attention and focusing stages need parallelization, they are assigned to FPGA. On the other hand, since the data to be processed is reduced down, high level tasks are performed in PC.

SRAM FPGA is known as power efficient comparing with Flash FPGA. SENTIOF-CAM [28] is an example of wireless vision sensor node with SRAM FPGA, proposed in 2014. Authors implemented a low-complexity background subtraction, segmentation and bi-level video codec in FPGA. They employed the system in particle detection, meter reading and people counting applications. In these applications, processing power is measured as 670 mW.

Utilizing off-the-shelf processing devices such as MCUs and FPGAs provide high flexibility, however ASIC solutions and transistor level optimizations are needed in order to reach much power efficiency by sacrificing this flexibility. Focal-plane image processing is one of the most efficient methods to reduce the power consumption. An example image sensor which has focal-processing array, FLIP-Q, is presented in [29]. Inside of each sensor area, pixel level analog processing elements operating concurrently with photosensing were incorporated. Having analog processing elements reduce the accuracy, but they are more area and power efficient. Resolution of the sensor was 176 × 144 fabricated in a 0.35 μm technology. It is reported that power consumption was maximum 5.6 mW. Later this platform was integrated to a WVSN node, Wi-FLIP [30].

Fill factor of an image sensor is the ratio between the light sensitive area to its total area and affects the sensitivity [31]. Placing the processing elements inside the pixel area reduce the fill factor, and hence the sensitivity as well. In order to overcome this issue, sensor, processing and memory planes were integrated vertically in [32]. Moreover, the resulting data bandwidth limitations of this solution was eliminated by fully parallel connections with high density of through-silicon-vias. Benefiting from this approach, a 3D vision chip VISCUBE which has different vision tasks stacked was presented. It had nearly 100% fill factor and able to reach very high frame rates up to 1000 fps. However, power consumption of the processing tiers could vary from order of 10 mW to 100 mW.

Another efficient transistor level technique to reduce the power consumption was near-threshold computing [33]. By using this method, reducing the power consumption down to even sub-mW levels by using energy-efficient MCUs is possible as presented in [34], [35]. This smart camera architecture exploited a low power contrast-based imager at 128 × 64 resolution and a quad-core ultra-low power processor (PULP) operates near-threshold. It also used content aware analog circuits to extract meaningful data from the sensor which sends the post-processed data to the processor. By this way, the sensor to processor bandwidth was reduced by 31 times.

Although near-threshold computing enables energy efficient integrated circuits, its reduction is also limited [36] and it is known that scaling down the transistor size does not bring a significant power advantage [37]. Therefore, future power reduction attempts are expected to be from different directions. A recent approach, Energy-Quality (EQ) Scalability was presented in 2017 [38]. It is based on the idea that, high quality tasks need high energy, and reducing the quality, where it is tolerable, can bring significant power reductions. This approach was applied on vision applications and an EQ scalable feature extraction accelerator EQSCALE was implemented in [39]. EQSCALE is assumed to be the first feature extractor that operates in sub-mW range at VGA resolution and 30 fps.

In this chapter, we presented examples of multiple and wireless smart camera systems among many of them. It can be seen that, there is not indeed a single unique solution to the problem. Since there are different design concerns and challenges in terms of power, area, bandwidth and flexibility, solutions also diversify. In the next chapters, inspiring from the ideas that were presented in the state-of-the art systems, we will develop the smart camera architecture targeting at a good balance between the system requirements and limitations.

3 Architecture

In this chapter, we present our smart camera architecture which is optimized in terms of resources and flexible enough to incorporate several processing modules with respect to the intended application. After giving an overview of the architecture, we will detail which functionalities that have been integrated as sub-modules into the proposed system.

3.1 Overview

Our smart camera concept consists of three key components is shown in Figure 3.1. An image sensor captures the frame and sends the pixel information with the synchronization signals. In fact, the most common method is to save the frame into a memory (*i.e. frame buffer*), and process this pixel data by accessing from there. Instead of this, our image analysis chip directly receives the data form the sensor, and performs its operations on the fly, benefiting from pixel buffers to reduce the output latency. The processing blocks which performs local analysis does not wait the end of the frame. Therefore metadata of a frame is produced during its stream.

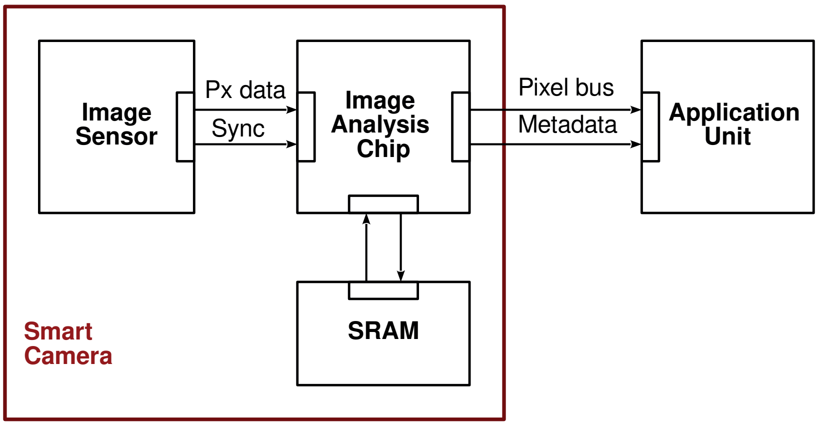


Figure 3.1 – Block diagram of the smart camera

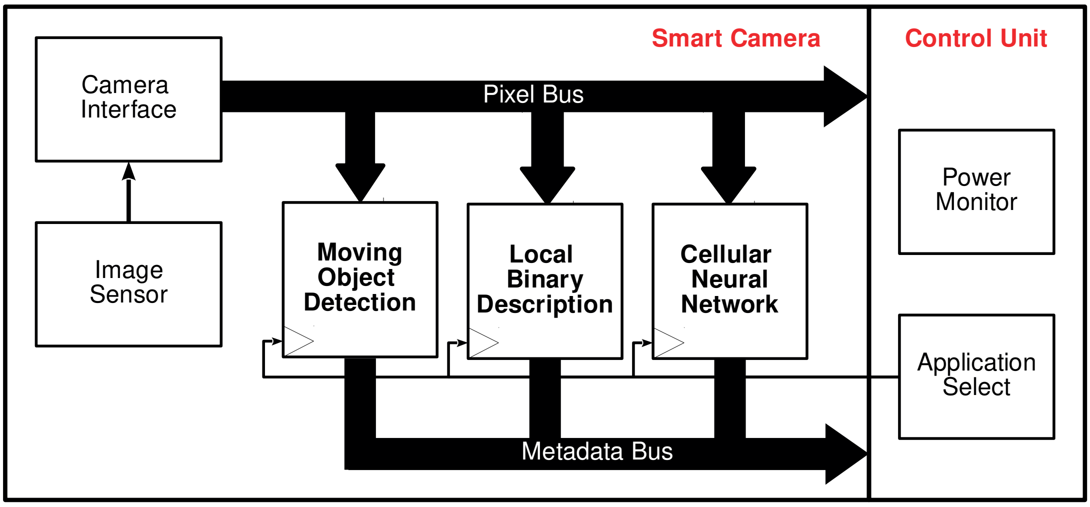


Figure 3.2 – Image analysis chip of the smart camera

A closer look to the image analysis chip is shown in Figure 3.2. Firstly a camera interface compatible with the image sensor is needed to receive the pixel data and synchronization signals, and convert them into pixel color information and pixel row/column addresses. Many computer vision algorithms process the gray level intensity, not the color. Since the implemented block in this architecture also process the grayscale data, an RGB to grayscale conversion block is embedded into the camera interface block.

Processing blocks which are connected to the pixel bus are activated by the application unit. The activated block drives the metadata bus, while the others have high impedence outputs. It processes the pixel stream on-the-fly and generates metadata.

The produced metadata is received by an application unit at lower bandwidth than the pixel stream. Thus, the application unit can achieve at better performance by processing this reduced amount of data. Each processing block produces metadata at different energy levels. In power critical systems, a decision mechanism which take into account the power level of the system observed by power monitor is needed. A list of possible metadata is listed below:

- Location and size of the moving objects
- Keypoint positions and their descriptions
- Classification of the objects
- Disparity - depth
- Polarimetry data

In this thesis, we show the implementation of three different application blocks, namely moving object detection, local binary description and cellular neural network, in the following sections.

3.2 Moving Object Detection

Applying resource-hungry complex analysis for each pixel at the each frame generally prevents one to meet real-time and power requirements. In fact the amount of the data to be processed must be reduced in order to keep resource requirements as low as possible. Most of the cases, pixel information remains the same from frame to frame, and a small portion shows a difference. However, those changes may not represent salient information and could belong to the movements out of interest, such as fountain, water flow, clouds, shaking of the leaves or grasses in natural images. Therefore, retrieval of the most relevant pixels in the foreground that bare the most important information should be targeted. If such relevant pixels could be detected, the system would simply ignore the rest where there is no significant change and this would speed up the processing, satisfying real-time requirements. The goal is accordingly to detect salient changes and avoid to re-compute the pixel positions that are part of either background or noisy and uninformative changes. This problem is well studied concept under the moving object detection title. In this section, we first present an overview of the moving object detection techniques and then propose a hardware-oriented scheme aiming at memory and power efficiency with reasonable accuracy.

Moving object detection problem has been often approached with background subtraction techniques. Accordingly, several methods have been proposed in the literature targeting different efficiency components: memory, power, accuracy, speed, etc, or providing a good trade-off between those concerns. Robustness of the method is also an important challenge. It is an expectation that the algorithm should handle illumination changes and filter the non-salient motions. We will mention some of such techniques including the ones that are more relevant to our study.

One of the common pixel stream methods is running Gaussian average method [40]. For each pixel position, it fits a Gaussian probability density function (pdf) of the last n pixels. But in order to prevent re-computation of the pdf for each frame, a running average is calculated as given below:

$$\mu_t = \alpha I_t + (1 - \alpha)\mu_{t-1} \tag{3.1}$$

where I_t is current value of the pixel, μ_t is the previous average, and α is an empirical value can stand for learning rate.

A pixel location at each frame can be labeled as foreground, if it meets the following inequality:

$$|I_t - \mu_t| > k\sigma_t \tag{3.2}$$

In the running Gaussian average approach, the background model is updated even if the pixel marked as foreground. In order to prevent this updates, a selective update method is proposed:

$$\mu_t = M\mu_{t-1} + (1-M)(\alpha I_t + (1-\alpha)\mu_{t-1} \tag{3.3}$$

where M is 1 if the pixel belongs to a foreground object, otherwise it is 0.

Adaptive Mixture of Gaussians (MoG) [41] is interested in adaptive modeling of the background for real-time tracking of scenes with complex and non-static backgrounds. They accordingly model each pixel as a pixel process with K Gaussians (K being 3 to 5) with corresponding mean and variance parameters and update the mixture model with each new pixel value over time. The background model is described by the first most probable distributions and a new pixel accordingly assigned as background if the most fitting Gaussian distribution to that pixel is deemed to belong to background model, as foreground otherwise. Each distribution represented by its mean and variance as floating point numbers.

The work in [42] proposed an hardware implementation of the MoG method by simplifying the formulations using look-up-tables (LUTs) which results in an FPGA-based circuit outperforming its counterparts as well as an ASIC baseline for further research. It is implemented on FPGA and ASIC UMC-90 nm process, and achieved 60 fps at full-HD resolution with 33.2 pJ/pixel power performance.

W_4 (Who? When? Where? What?) is a widely used method in surveillance systems [43] In this method, background model is compound of minimum (μ_{min}), maximum (μ_{max}) and largest absolute difference (D_{max}). A pixel at a position i, j is determined to be foreground if it satisfies one of the following inequalities:

$$|I(i,j) - \mu_{min}(i,j)| > D(i,j) \qquad \text{or} \qquad |I(i,j) - \mu_{max}(i,j)| > D(i,j) \tag{3.4}$$

There are also some methods which includes several methods at the same time. In [44], authors benefit from both frame difference and W_4 methods. This two approach is implemented seperately and the outputs are combined with a logical `OR` operation.

One of the recent background subtraction algorithms for embedded vision is EBSCam [45]. This method is based on the suppression of the variation of the background mode. It provides a competitive accuracy with better hardware efficiency in terms of both area and speed. In this method, for each pixel location 90 bits need to be stored.

A memory-efficient and light-weight foreground detection method is presented in [46]. Although this method needs less memory for each pixel, it is robust against lighting variations and non-static background. Method selectively updates the background model with an automatically adaptive rate, thus can adapt to rapid changes. Pixels are not treated individually, and information about the neighbours is incorporated into decision making. Instead of building a mathematical model for each pixel location, this method observes the state changes being

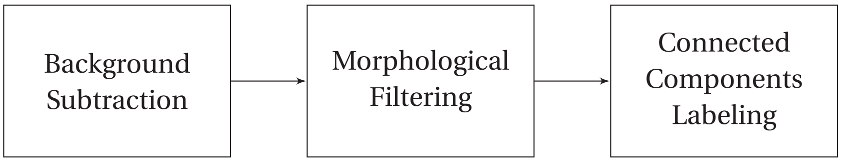


Figure 3.3 – Moving object detection pipeline

between background and foreground in the pixel position and its neighbours. If the number of state changes exceeds a threshold, that pixel marked as a non-reliable pixel. For the foreground pixel decision, reliability of the pixel and its neighbours are taken into consideration.

Background subtraction produces a binary image shows the foreground pixels. This binary data can be processed by a connected component labeling block, in order to group the pixels which belongs the same object. Although there are many efficient solutions of the connected component labeling problem in software, their iterative process needs to access the binary image several times. Moreover, at least one binary frame must be stored in the memory. Thee Hoshen-Kopelman algorithm [47] applies two passes over the image. In the first pass, it assigns incremental numbers as temporary labels and finds the equivalent pixels. In the following second pass, it replaces each temporary label by the smallest label of its equivalence class.

More hardware friendly approaches are also presented in the literature. In [48], authors propose a real-time blob analysis methods by using only one single pass, and implement it on the FPGA. Before the operation, the pixels are compressed by a run length coding. A linked list based FPGA implementation is [49]. Another example of the single-pass methods is presented in [50] and [51]. This method requires less memory than the alternatives, and can reach high throughput.

Since our smart camera concept utilizes an external vision sensor chip, sensor level processing is not an option. Instead, we target to have a with a low complexity design by processing the pixel stream reducing the memory bandwidth and storage requirements. Therefore, similar to [46], we compute the number of switches between background and foreground for each pixel, to determine its reliability. Foreground detection takes into account the reliability information of the pixel and its neighbours. In addition to this, thanks to hardware centric approach and pixel processing on-the-fly, we reach high accuracy results with very low memory and power requirements. Moreover, we incorporate the background subtraction block with a morphological filtering and connected component labeling blocks, to give clean results ready to be processed.

This moving object detection block is connected to the pixel bus, and when a new pixel is received, its data directly feed the processing pipeline. As presented in Figure 3.3, the moving object detection block constitutes of background subtraction, morphologic filtering MF and connected components labeling blocks. Background subtraction block creates a binary image from the difference of the current frame and the background model. This binary result directly feeds a shift register of the morphologic filter which removes the shapes of dimensions smaller than 2 × 2. Non-filtered binary values are evaluated by the connected component labeling block. It determines if a pixel belongs to a previously found object or a new one and accordingly updates the results. The output of the moving object detection process gives the location of the objects and their boundary boxes.

This integrated method creates the metadata which is the positions and dimensions of the moving objects. In total, it requires 18 bits per pixel for background model, and 5 binary line buffers. Since it is based on the on-the-fly processing, the latency is minimum. As soon as a moving object is detected, its information is immediately sent to the metadata bus without waiting end of the frame. In the following subsections, we explain the methods in the pipeline in detail.

3.2.1 Background Subtraction

In the background subtraction block, two main operations are performed: (1) creating and updating the background model, (2) evaluation of the pixels as background or foreground. This two operations changes depending on the information level of the system. From the initialization signal, states of the frame can be illustrated in Figure 3.4. Normally, this hardware remains in IDLE state when the operation is not necessary, or lack of power. With a trigger signal, it switches to INIT state where the first captured frame is set as the background for fast adaptation:

$$\mu_t(i,j) = I_t(i,j) \tag{3.5}$$

where $\mu_t(i,j)$ is background model for the pixel location i,j, and $I_t(i,j)$ is gray level pixel intensity of the captured frame. After the first frame, absolute value of the difference of the intensity of a pixel position (i,j) and the value at the background model in the same pixel position is:

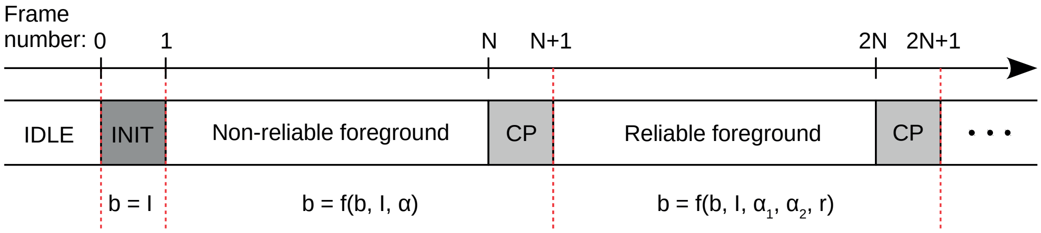


Figure 3.4 – Frame states for background subtraction block

$$d = |I_t(i,j) - \mu_{t-1}(i,j)| \tag{3.6}$$

If the difference d is smaller than a threshold θ_1, it means that there is not a significant change in the background model, and the background does not updated. It prevents unnecessary memory access and switches, thus power consumption is reduced down. Otherwise, it is updated by running Gaussian average:

$$\mu_t(i,j) = \begin{cases} \mu_{t-1}(i,j), & \text{if } d < \theta_1 \text{ or } d > \theta_2 \\ \alpha\mu_{t-1}(i,j) + (1 - \alpha I(i,j)), & \text{otherwise} \end{cases} \tag{3.7}$$

where α is an ampirical pre-defined learning rate, and selected as 2^{-N} for hardware efficiency. If this difference d exceeds a threshold θ_2, status of the pixel is determined as foreground:

$$s = \begin{cases} 1, & \text{if } d > \theta_2 \\ 0, & \text{otherwise} \end{cases} \tag{3.8}$$

In this state, there is no reliability information. Therefore foreground output will always be 0. But this information is used to compute the number of the state changes of each pixel:

$$r_{c,t}(i,j) = \begin{cases} r_{c,t-1} + 1, & \text{if } s_t(i,j) \oplus s_{t-1}(i,j) \\ r_{c,t-1}, & \text{otherwise} \end{cases} \tag{3.9}$$

where $r_{c,t}(i,j)$ is a register which is incremented if there is a state change. A logical *XOR* operation applied on current and previous states gives the state change, and in this case state switch register is incremented by one.

When the frame counter reaches N, the system switches to checkpoint state and decides the reliability of the pixels. If the number of state switches exceeds the threshold θ_r, then it is determined as it swithces too much and does not give a reliable information:

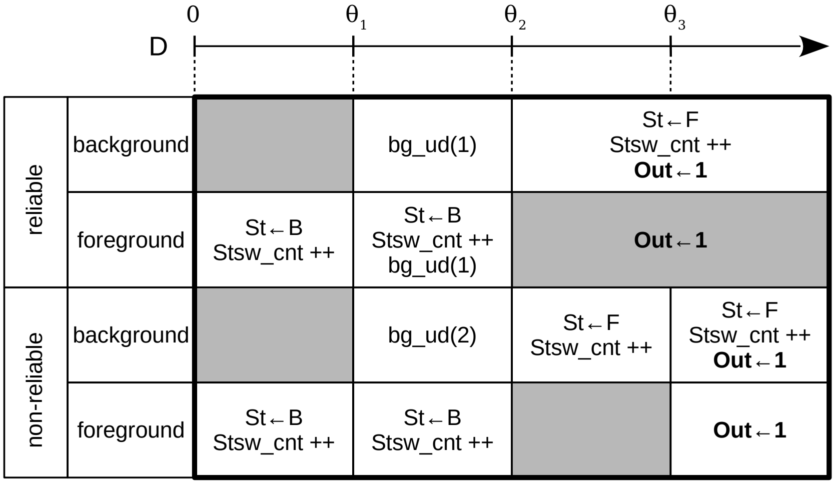
$$r(i,j) = \begin{cases} 1, & \text{if } r_c(i,j) < \theta_r \\ 0, & \text{otherwise} \end{cases} \tag{3.10}$$

where r stands for reliability bit of the pixel. System resets the register which keeps the number of state switches r_c. The next N frames are evaluated according to this reliability information.

After processing the first checkpoint frame, having the reliability data of the pixels enable more accurate background model and foreground information. The evaluation operation is summarized in Figure 3.5. In this state, background update is a function of the difference d, and the reliability r.

$$\mu_t = \begin{cases} \mu_{t-1}, & \text{if } d < \theta_1 \\ \alpha_1\mu_{t-1} + (1-\alpha_1)\mu_{t-1}, & \text{if } d > \theta_1 \text{ and } r = 1 \\ \alpha_2\mu_{t-1} + (1-\alpha_2)\mu_{t-1}, & \text{if } d > \theta_1 \text{ and } r = 0 \end{cases} \tag{3.11}$$

where α_1 and α_2 are update rates such that $\alpha_1 > \alpha_2$. By this way, small changes in reliable pixel positions are updated faster than the non-reliable pixel positions, because the confidence



reliable	background	▨	bg_ud(1)	St←F Stsw_cnt ++ **Out←1**	
	foreground	St←B Stsw_cnt ++	St←B Stsw_cnt ++ bg_ud(1)	▨ **Out←1**	
non-reliable	background	▨	bg_ud(2)	St←F Stsw_cnt ++	St←F Stsw_cnt ++ **Out←1**
	foreground	St←B Stsw_cnt ++	St←B Stsw_cnt ++	▨	**Out←1**

Figure 3.5 – Advanced evaluation of the pixels

level of the coming pixel is higher.

For a reliable pixel position, if the difference exceeds the threshold θ_2, state output is 1. However, for a non-reliable pixel position, it is already expected to oscillate background and foreground. Therefore, the system filters this change. In this case, in the non-reliable areas the output will be masked, even if there is a significant abnormal change. In order to cover this case, we define another threshold θ_3, to detect the movements in non-reliable areas. This threshold is determined during the first non-reliable foreground operation, by storing the maximum difference for each pixel position. Output s_{out} is given in the following equation:

$$s_{out} = r.(d > \theta_2) + \overline{r}.(d > \theta_3) \tag{3.12}$$

One of the main target of this method is keeping the memory allocation and access minimum. To reach this goal, we store 18-bit words for each pixel as shown in Figure 3.6. Most of the SRAMs in the market is designed as 9, 18 or 36 bit words, therefore, 18 bit selection suits well the available memories. The first 8 bits keep the background model of the pixel. This data is used to compute the difference d in the Equation 3.6 The next 5 bits stores the number of switching activities. Reliability and status occupy one bit each. The final 3 bits are dedicated to maximum difference for non-reliable pixels. Before each incoming pixel captured from the pixel bus, its information must be requested from the memory and be ready for the evaluation process. After the evaluation, pixel data on the memory may be updated. However, if the information stored in the memory location does not change, then it is not needed to update the information. The situations which does not require a memory write request is marked as gray boxes in Figure 3.6.

Sometimes a moving a foreground object appears in the scene but then it stops and keep its position for a long time. After a point, it is better to evaluate this pixel as a part of background.

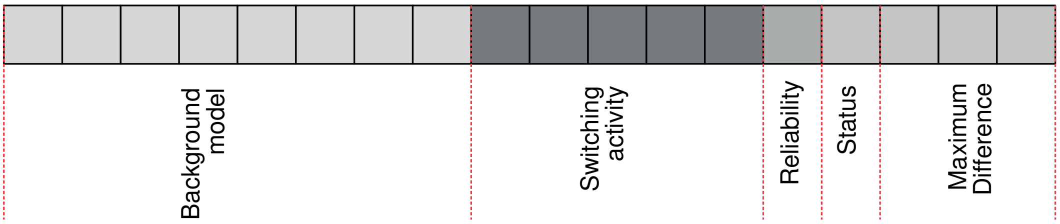


Figure 3.6 – Content of a memory row stores information of one pixel

Because, this moving object information is already produced and sent to the application unit. Re-computation of this area creates unnecessary signal switches without a useful information. Therefore, we update the background with a small update rate.

This background subtraction method brings several advantages in terms of energy consumption. We can formulate the total energy requirement of a general background subtraction operation based on the frame difference method as in the following:

$$E_{bs} = E_{op} + E_{cw} + E_{cr} + E_{bw} + E_{br} \tag{3.13}$$

where E_{cw} and E_{cr} are the energy consumption required to write and read the current frame, while E_{bw} and E_{br} are the ones to update/write and read the background model respectively and E_{op} is the energy used for the computation. In our case, since the pixel stream is evaluated on-the-fly, the energy consumption due to read and write operations, i.e. E_{cw} and E_{cr} are eliminated. Moreover, E_{bw} involves only when the background is updated. For most of the cases, E_{bw} does not contribute which also reduces the total energy consumption.

Another important advantage is reduced memory space. While the memory requirement per pixel in average more than 300 bits in [41], and more than 400 bits in [52]. In the work [46] which is also our starting point is needs to store 2 integer number in addition to pixel data. Our hardware oriented approximation lead us to achieve 18 bits per pixel and background data for each pixel can be stored in a single cell of 18-bit configurations of the SRAMs.

3.2.2 Morphologic Filter

Background subtraction module produces a binary output stream, and this output may contain high amount of salt-and-pepper noise. This noise increase the workload of the connected components labeling block, and cause a memory and power penalty. In order to remove such noise, morphological erosion, an operation performed on binary images, is widely used.

Morphologic operations can also be defined as the convolution on binary images. In this case, in a convolution kernel the following operation is applied:

$$P(i,j) = \prod_{k=i-n}^{i+n} \prod_{l=j-m}^{j+m} P(k,l) \tag{3.14}$$

Erosion operation can be applied by a sliding window of 2 × 2 size as depicted in Figure 3.7. In this case, kernel of this convolution is all ones:

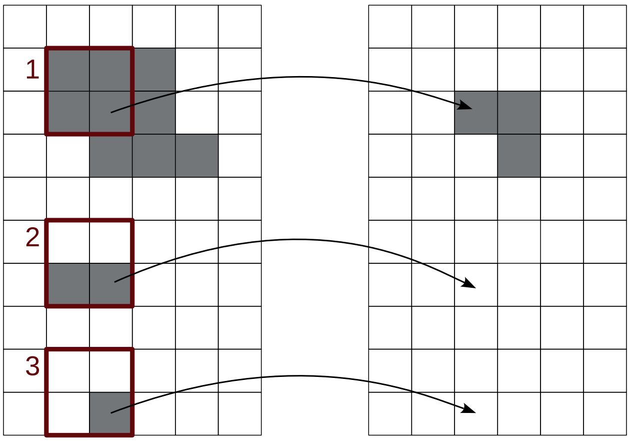


Figure 3.7 – Example operation of a 2 × 2 erosion filter

$$K = \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix}$$

If all the pixels in the window are 1, then bottom right corner position in corresponding pixel location in the output image is also 1, otherwise, it is 0. Thus, the first window produces 1, while the other two positions 0. By this way, foreground pixel groups consists of high number of connected pixels still remains in the result, small pixel groups smaller then 2 × 2 disappears. Although this non-linear transformation does not preserve the shape of the areas, presence of the large object can be detected.

Hardware implementation of a 2 × 2 erosion filter which removes components of dimensions smaller than the applied filter size is shown in Figure 3.8. In this hardware, binary input generated by background subtraction block is provided to the shift register. The erosion operation is performed using logic AND operation, between the currently evaluated pixel and its pre-determined horizontal or vertical neighbors.

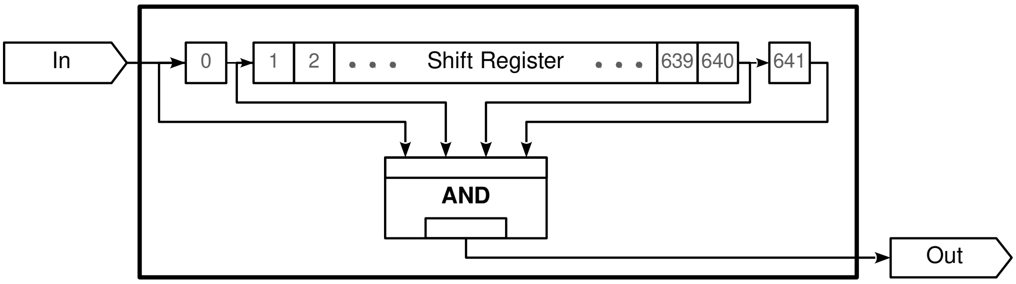


Figure 3.8 – Hardware implementation of the erosion filter

3.2.3 Connected Components Labeling

The binary output produced by foreground detection block and filtered by morphology operation finally arrives at the connected components labeling block. This operation assigns different labels to the pixels groups which are constituted of the connected pixels. In other words, a pixel is assumed to belong to a pixel set, if it either touches or it is very close in a pre-defined distance. Connected components labeling can be done in an iterative manner, as in many software implementations, however iteration brings the requirement of multiple accesses to the image, which requires more memory and energy, and causes latency. Instead of iterative methods, binary data can be evaluated as soon as the filtered output is received by the connected components labeling block in the hardware, hence memory storage and access requirements are eliminated.

In this part, we propose a hardware oriented connected components labeling algorithm, which computes the data on-the-fly, compatible with the stream of the pixels in the raster-graphics format. Evaluation of the pixel is done in a single clock cycle and the hardware outputs the position and dimensions of the frames which covers the connected pixels to the metadata bus during the pixel stream. In order to achieve this performance, we sacrifice an amount of accuracy, as stated in energy-quality scalable design manner. As a consequence of this quality degradation it may fail to separate the close but different objects. However, in case of the accuracy is not very critical, *i.e.* if the approximate number of the moving objects is important, it provides enough accuracy with very low cost of hardware.

Data structures and control register

In this design, dimensions of an instance is represented by 8 numbers as illustrated in Figure 3.9. Four parameters define the rectangular boundaries of the instance as x_{min}, x_{max}, y_{min} and y_{max}, and constitutes of vector $\vec{A}$.

x_0 and x_1 stores the bottom left and right corner positions of the last row of the active object. This two parameters are used to determine whether the newcoming foreground pixel touches

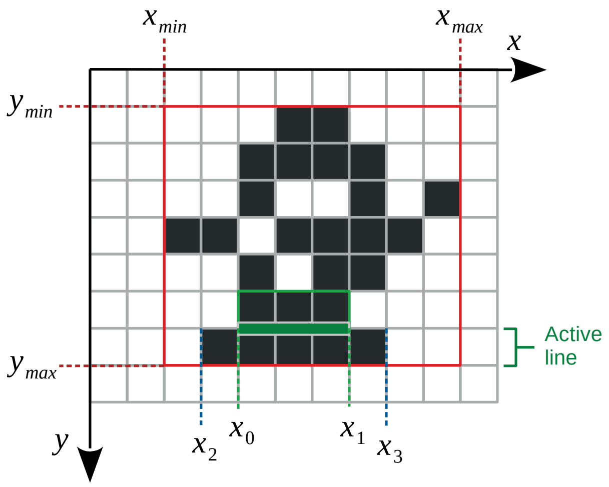


Figure 3.9 – Data structure representing an instance

to that object or not. The column addresses of the left-most and right-most foreground pixels which belongs to the same object are stored in x_2 and x_3, respectively. When the row is completed in the pixel stream, x_2 and x_3 will be the new boundaries and replaces x_0 and x_1 in the next row. This vector consists of x_0, x_1, x_2 and x_3 is defined as vector $\vec{B}$. Thus, a moving object is represented by two vectors: $\vec{A}$ and $\vec{B}$ which corresponds the boundaries and the bottom line information, respectively. All the operations can be explained by using this data structure, composition of $\vec{A}$ and $\vec{B}$.

In addition to this data structures, a control register stores the states of the objects in 2 bits per instance. One of the registers stores if the instance is active, *i.e.*, its bottom line has pixel(s) from the previous row. In other words, if there is a possibility of newcoming pixel touching an instance, which results in updating or merging operations on it, this instance is marked as active. The second register stores if an instance has new pixel during the stream of the current row. Considering these two registers, if an instance was active but did not receive any new pixel in the active row, it means that the instance is completed and its information is ready to be sent through the metadata bus.

Since the number of the instances is fixed at 8, then the size of the control register is determined as 16-bits. If we use 10-bit register for each element of $\vec{A}$ and $\vec{B}$, 80 bits are dedicated for each vector, and 640 registers in total. To sum up, the 656 flip-flops are used for instance data structures and control register.

Operation

Operation of the CCL block is managed by an FSM which is shown in Figure 3.10. This FSM has 6 states, namely:

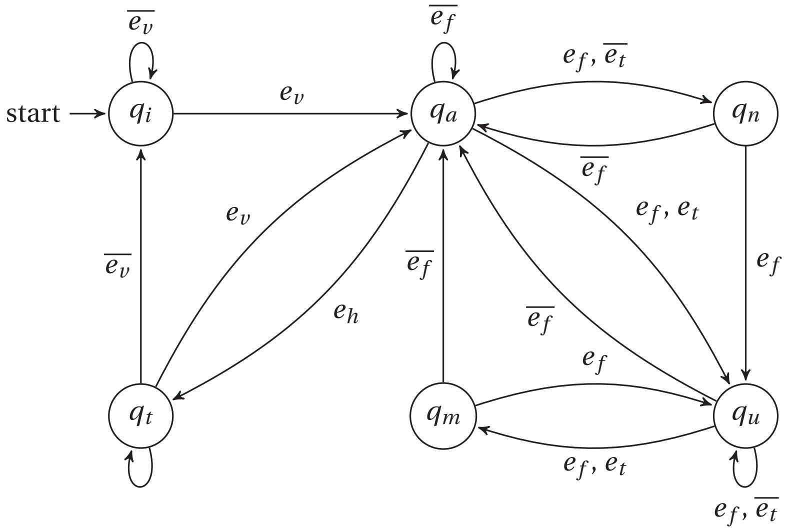


Figure 3.10 – FSM for connected component labeling

- q_i: init state
- q_a: active waiting state
- q_n: new instance state
- q_u: update instance state
- q_m: merge state
- q_t: transmit state

Transactions between the states are triggered by the following events:

- e_h: end of line (horizontal blanking)
- e_v: frame start (vertical blanking)
- e_f: receiving a foreground pixel
- e_t: the new foreground pixel touches an instance

The FSM starts from q_i state, and waits for a frame start signal e_v to switch to q_a state. When a foreground pixel arrives, if it touches to any available objects, FSM switches to the update state q_u. Otherwise, it goes to q_n state to create a new instance. In q_n state, it creates a new instance with the following parameters:

$$A_{new} = \begin{bmatrix} x_{min} \\ x_{max} \\ y_{min} \\ y_{max} \end{bmatrix} = \begin{bmatrix} x \\ x \\ y \\ y \end{bmatrix} \qquad B_{new} = \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} x \\ x \\ x \\ x \end{bmatrix} \tag{3.16}$$

where x and y is the coordinates of the new foreground pixel. In q_n state, if the new input is foreground, it goes to q_u state to update the current instance:

$$\vec{A} = \begin{bmatrix} x_{min} \\ x_{max} \\ y_{min} \\ y_{max} \end{bmatrix} = \begin{bmatrix} \min(x, x_{min}) \\ \max(x, x_{max}) \\ y_{min} \\ y \end{bmatrix} \qquad \vec{B} = \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x \end{bmatrix} \tag{3.17}$$

While it is in the q_u state, if it touches another instance, FSM enters to the merge state q_m. In this state, current active instance vector is updated with the information of the touched instance as in the following equations:

$$\vec{A} = \begin{bmatrix} x_{min} \\ x_{max} \\ y_{min} \\ y_{max} \end{bmatrix} = \begin{bmatrix} \min(x_{min}, x'_{min}) \\ \max(x_{max}, x'_{max}) \\ \min(y_{min}, y'_{min}) \\ y_{max} \end{bmatrix} \qquad \vec{B} = \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} x_0 \\ x'_1 \\ x_2 \\ x \end{bmatrix} \tag{3.18}$$

where the touched instance is represented by $\vec{A} = x'_{min}, x'_{max}, y'_{min}, y'_{max}$ and $\vec{B} = x'_0, x'_1, x'_2, x'_3$. Removing of this instance is simply done by resetting the corresponding activation bit of the instance in the control register.

Example

A comprehensive example is illustrated in Figure 3.11. In this example, sixth line is the active line, *i.e.*, the connected component labeling block is receiving the information regarding to the sixth line. The instance #1 is a passive object which is already sent and there is no connection with the active line. Therefore receiving pixel cannot belong to this instance which makes it passive.

Instance #2, #3 and #4 are active instances that are possible the foreground pixel can touch. $\vec{A_2} = (5, 10, 1, 5)$ and $\vec{B_2} = (6, 9, -1, -1)$ represents the second instance, $\vec{A_3} = (12, 14, 2, 5)$ and $\vec{B_3} = (13, 14, -1, -1)$ represents the third instance. At the beginning of the line, FSM is in q_a state. When the first foreground bit arrives at $x = 3$, it changes the state to q_n to create a new instance #5. FSM switches to the update state f_u with the next foreground bit. The third

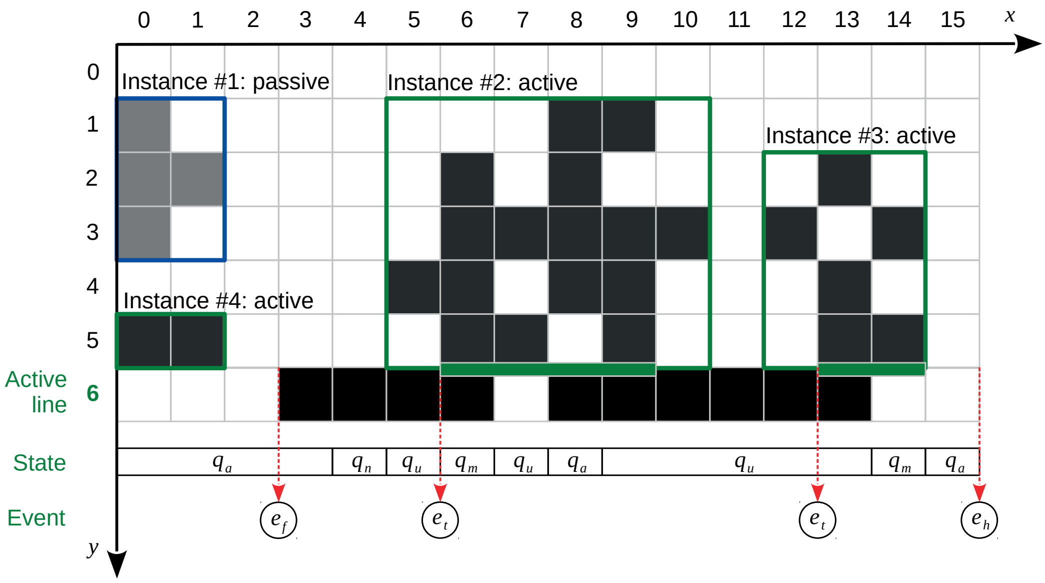


Figure 3.11 – Connected components labeling example operation

foreground bit at $x = 5$ creates the touch event e_t, and FSM switches to the merge state q_m to combine the instances #2 and #5. The instance #5 is updated with respect to the Equation 3.18, and the new values for $\vec{A}_5$ and $\vec{B}_5$ are:

$$\vec{A}_5 = (3,10,1,6) \qquad \vec{B}_5 = (6,9,3,5) \tag{3.19}$$

The next foreground pixel changes the state again q_u, and the following background pixel to q_a. Then a string of 6 consecutive foreground pixels arrive. For each foreground pixel, FSM updates the instance #2 in the q_u state, until it reaches $x = 12$. At this point, touch event occurs to the instance #3, and the FSM switches again to the merge state.

$$\vec{A}_4 = (3,14,1,6) \qquad \vec{B}_4 = (6,14,3,13) \tag{3.20}$$

At the end of the line, it enters the q_t state where it transmits the completed instances, and update the active instances for the next line. In this example, since although instance #4 is active, there is no foreground pixels touches to it in the active line. This makes it completed $\vec{A}_4$ is transmitted via the metadata bus:

$$\vec{A}_4 = (0,1,5,5) \tag{3.21}$$

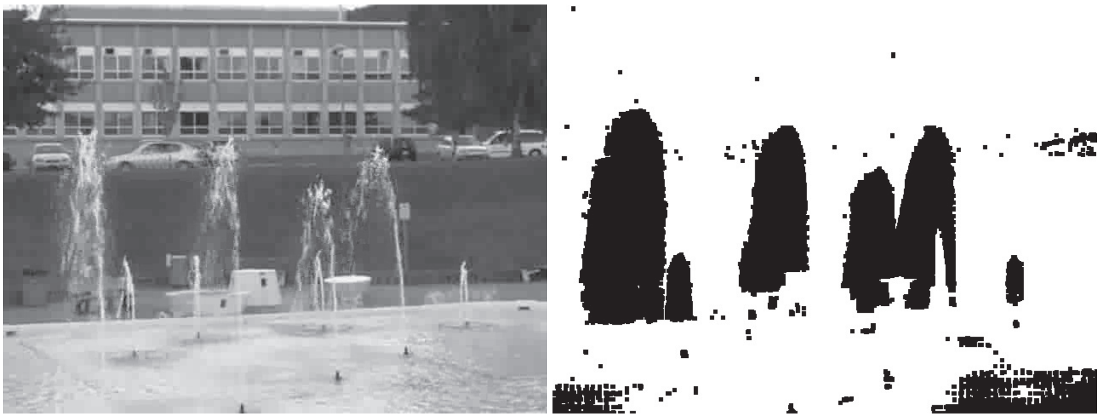
One of the limitations of the hardware, in order to achieve one clock cycle timing performance, it uses a static number of instance positions, which is 8 in this implementation. It means that, there cannot be more than 8 active instances at a given time. However, this is not a limitation of the maximum instances can be detected in a frame. Because, if an instance does not have a connection in the active line anymore, its data is sent during the horizontal blanking.

3.2.4 Result

We tested our method with the change detection dataset from [53], and compared its results with OpenCV implementation of Mixture of Gaussians (MOG). In general, MOG is a more robust method which covers more cases. However, in some challaeging frame sets, especially where there are non-salient motion such as shaking of leaves or fountain, our method outperforms the MOG method in terms of accuracy.

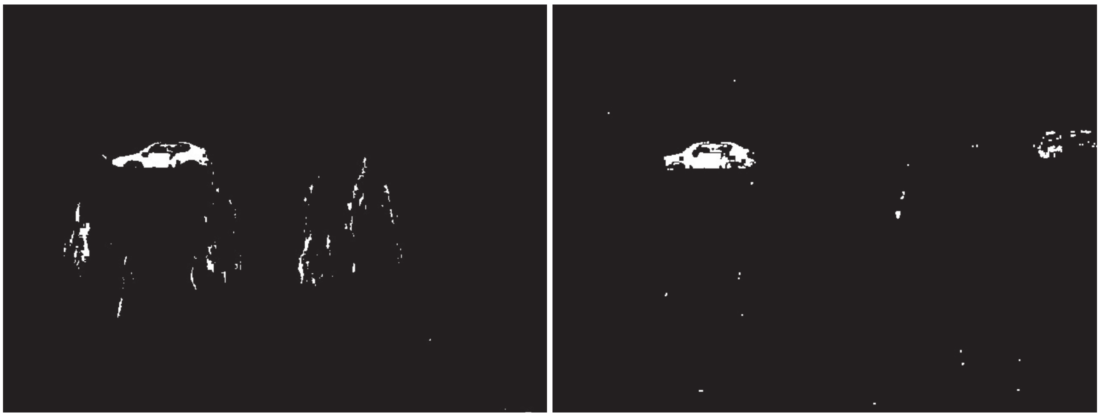
Result of such a frame set is shown in Figure 3.12. Figure 3.12a is an example frame which has a moving car and fountain which creates non-salient foreground data. Our method observes the changes, and creates a binary reliability matrix for the frame. This matrix is used to mask non-salient data, *i.e.*, fountain region in this case. As shown in Figure 3.12c and Figure 3.12d, our method gives much cleaner result, by filtering the fountain region.

On the other hand, considering memory efficiency, timing performance and power consumption, our method well suits the resource limited embedded devices.



(a) Frame

(b) Reliability



(c) OpenCV MOG result

(d) Output of our method

Figure 3.12 – Foreground detection results of the proposed method

3.3 Feature Extraction

Extracting the salient features is one of the most common operations in computer vision applications. These features carry the significant portion of the information about the image. Therefore, eliminating the remaining redundant data and perform computations on the features provides an efficient analysis. This approach is widely used in many applications including object recognition, motion tracking and image registration.

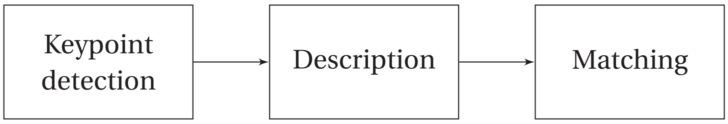


Figure 3.13 – Feature extraction pipeline

A classical pipeline for feature extraction process is shown in Figure 3.13. Firstly, image pixels are tested according to selected feature detection criteria, and feature positions are transferred to the description block. For each detected feature, its description is computed with a selected algorithm. The final block receives the description results, and find the best matches.

A feature detection or description algorithm is expected to be robust to image transformations. From the images taken from different viewpoints in the same scene, under rotation or scaling conditions, they should find similar keypoints, and close descriptions. In addition, these features should be distinctive to find the correct matches. Moreover, accuracy is not the only concern in embedded systems. In addition, power, area, memory and timing efficiencies are also critical.

An important feature detector available in the literature is Canny Edge Detector [54]. Edge detection problems have been formulated considering three criterion: good detection, good localization and elimination of spurious detections. Such criteria have led to a 4-step algorithm as follows: (1) Apply Gaussian filter to smooth the image (2) Compute the intensity of the gradient image (3) Non-maximum suppression to remove irrelevant (noisy) responses to edge detections (4) Use of double-threshold and edge tracking by hysteresis. Canny Edge Detector has been widely used in computer vision applications due to its easy implementation and good and reliable performance.

Another popular feature detection technique is Harris corner detector [55]. This technique aims at detecting both edges and corners simultaneously, claiming that edge detectors such as Canny Edge Detector do not consider edge connectivity while performing edge detection and that detecting corner, as defined the junctions of edges is indeed very crucial for edge detection. As an improvement of Morovec's corner detector [56], the basic idea behind the proposed technique is to consider three potential cases to discriminate salient parts from non-interesting regions: The intensity in a small window (1) does not change in a flat region when shifted to any direction (2) changes only in the edge direction when on an edge (3) changes in all directions when on a corner. In [55], the authors formulate those three cases in one formulation and proposes to first compute the structure tensor H, second moment matrix for every pixel (x,y) of the sum-of-squared distances (SSD) between a small window around a point of interest and its shifted versions and second to calculate the corner/edge response function by

$$R(x, y) = det(H) - k Trace(H) \tag{3.22}$$

with k being an empirical parameter in [0.04, 0.06]. The response R is then thresholded to determine the flat regions, edges and corners.

In order to further increase the speed of feature detection up to a real-time scale, Rosten and Drummond [57] have proposed a technique for corner detection that examines a circle of 16 pixels around a candidate point. If a contiguous arc of 12 pixels that are either brighter or darker than the candidate point intensity is found, it is considered as a corner. As a further improvement, in [58], the authors apply a machine learning approach to determine how many pixels have to be contiguous around the candidate point that maximizes the information gain introducing a recursive decision tree-based algorithm. In the experiments, FAST algorithm seeking for an arc of contiguous 9 pixels around a candidate points was found to lead to the optimal performance than the previous assumption of 12 contiguous pixels.

Depending on scale of the image, keypoint property can differ. For example a corner point in a small kernel can be seen as an edge, while it is detected as a corner within a bigger kernel. SIFT (Scale Invariant Feature Transform) [59] propose the scale invariance for both detection and description. Its primary operation is scale-space filtering with Laplacian of Gaussian (LoG). Gaussian kernel with low σ gives high values for small corners, while high σ for large corners. Thus local maxima across the scale and space gives a list of (x, y, σ) values represents a keypoint at pixel position x, y at scale of σ. Since LoG is a costly operation, SIFT algorithm uses an approximation, Difference of Gaussians (DoG).

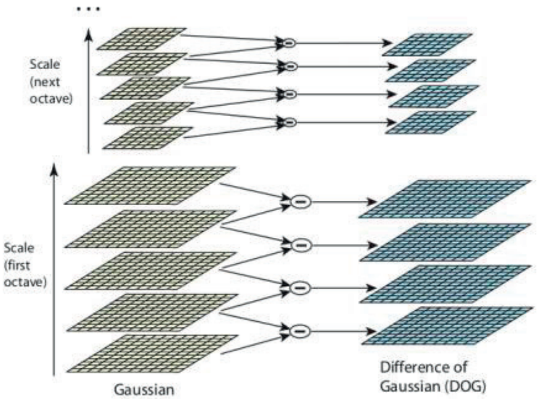


Figure 3.14 – Difference of Gaussians in SIFT [60]

SIFT is a highly discriminative and robust to many image transformations. However its

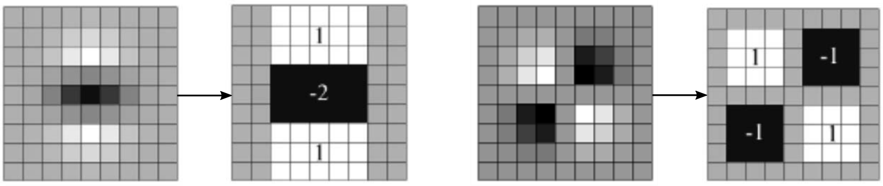


Figure 3.15 – Square box approximation in SURF [61]

computation requires gradient histogram pooling, and it produces a high-dimensional vector of floating point numbers. Therefore, cost of its operation is high in terms of memory and computational power. In order to reduce the complexity of the SIFT method, SURF (Speeded Up Robust Features) [61] is proposed. Instead of gradient histogram pooling, it uses Haar wavelet responses, and reduces the resulting vector dimensions from 128 to 64.

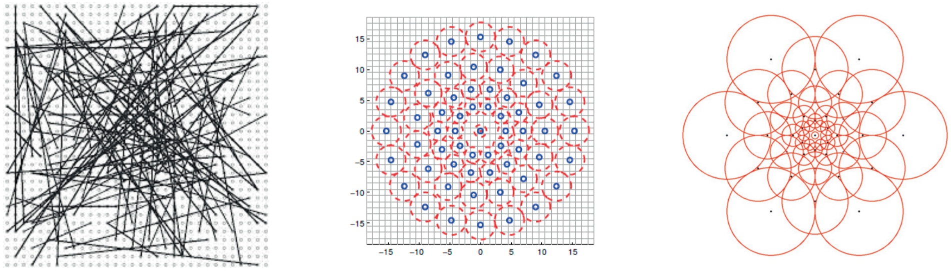
In SURF, instead of Gaussian derivatives, box filter approach is used as shown in Figure3.15. This approach makes it possible to benefit from the integral image, to compute the response of the filter fast and with much less resources. Similar to SIFT, it searches interest point in scale-space, but instead of building several layers of the image, it uses different size filters to search for points at different scales.

$$\mathscr{H}(p,\sigma) = \begin{bmatrix} L_{xx}(p,\sigma) & L_{xy}(p,\sigma) \\ L_{yx}(p,\sigma) & L_{yy}(p,\sigma) \end{bmatrix}$$

Even though SURF reduce the vector dimensions, the type of the elements of resulting vector is still floating point. Memory footprint of a description vector with 64 elements is 256 bytes, and in a frame with 1024 keypoints it needs 256 kBytes to store only the keypoint vector. Moreover, matching of the keypoints performed on floating points is a costly operation. Therefore, limited resources of the embedded systems usually are not able to satisfy their requirements.

In order to overcome this issues, BRIEF [62] method is proposed to be a binary descriptor, producing binary vectors for each keypoint. This method has two primary operations: (1) smoothing with a kernel to reduce the sensitivity and (2) intensity comparison between selected pairs. Experiments show that random or non-ordered selection of the pairs achieves better performance. An example of selection of the comparison pairs is given in Figure 3.16a. The output description result is a 512-bit binary array. This means a drastically improvement (from 256 kBytes to 0.5 kBytes) in memory efficiency. Moreover, matching of the keypoints are much faster. In order to find the similarity between 2 keypoints, Hamming distance which is a bitwise XOR operation followed by a bit count, is computed. In the BRIEF method, smoothing circles have a fixed diameter.

Binary Robust Invariant Scalable Keypoints (BRISK) [63] method is another binary description method. Differently from the BRIEF method, BRISK uses a regular sampling pattern with



(a) BRIEF [62]

(b) BRISK [63]

(c) FREAK [64]

Figure 3.16 – Comparison points in binary descriptor

a uniform density around the keypoint as shown in Figure 3.16b. In addition, BRISK needs dramatically fewer sampling points since a single point is used in different comparisons.

Inspiring from the increasing density of the neurons in the human retina from outer regions to the center, Fast Retina Keypoint (FREAK) [64] algorithm is proposed. Diameter size of the smoothing circles increases according to radial distance from the keypoint location as depicted in Figure 3.16c. This bio-inspired radial growth and overlapping of the circles in the FREAK method has been reported to be more effective in keypoint description [64].

Finding the descriptions of all the points is very time-consuming and power-inefficient, therefore description methods are incorporated with a keypoint detector. FAST corner detection algorithm [58] is commonly preferred due to its speed advantage. Although the binary description techniques and FAST corner detection pair have good computational performance among the others, their software implementations are still computationally expensive for real-time embedded applications. Yet, hardware implementations accelerate the computation and reduces the power consumption.

In [65], a method for detection and description as well as matching is proposed and implemented on a Zynq-based platform. They implemented the combination of the FAST and the BRIEF algorithms. A standalone FREAK implementation as a Xilinx IP core is proposed in [66]. There are also implementations of the ORB algorithm [67], which is a combination of FAST and BRIEF methods. In [68], authors implemented SURF method for feature detection, and FREAK for description on FPGA. They reach 60 fps frame rate for 800 × 600 resolution. FAST-BRIEF and FAST-BRISK implementations on different platforms such as Embedded CPU, Tegra GPU and Zynq are compared in [69]. Better performance can be reached by ASIC design. For instance, 135-frames/s 1080p 87.5-mW binary-descriptor-based image feature extraction is proposed in [70] with implementation in 65-nm CMOS technology.

In this design, we selected FAST [57] and FREAK [64] methods for keypoint detection and local binary description, due to their speed and memory efficiency. Since reducing the latency is one of our primary concerns, performing the computations on the pixel stream is crucial. In order to prevent unnecessary memory access which causes latency and power penalty, all

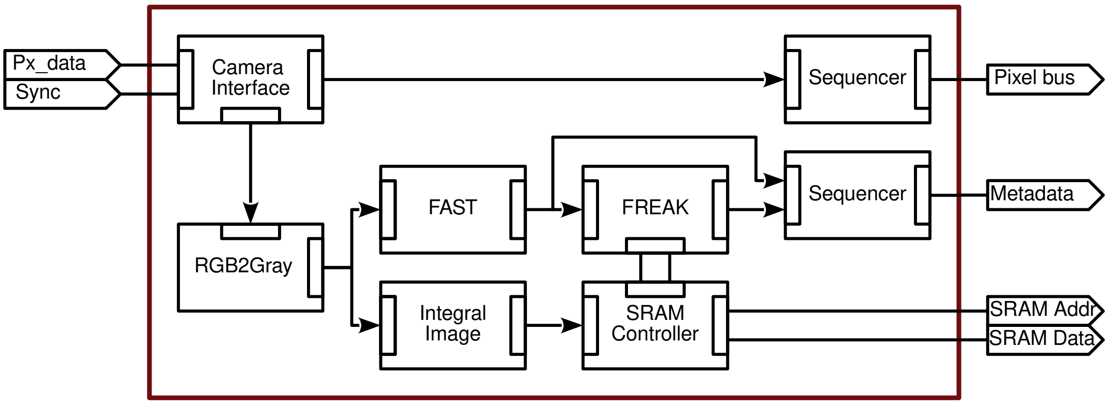


Figure 3.17 – Hardware block diagram for local binary description

computations must be progressed with the capture of the pixel data stream.

An overview of the hardware block diagram is shown in Figure 3.17. Resolved RGB pixel values and their addresses by the camera interface are firstly converted into grayscale values and then it feeds both the integral image and the keypoint detection blocks. While the integral image data is written onto SRAM via the arbiter, FAST block computes in parallel the location of the corners from the pixel stream. The resulting locations are sent to the FREAK description block. Since the FREAK operation requires multiple clock cycles per keypoint, a FIFO buffer is utilized to handle the keypoint coordinate transfer between the FAST and FREAK modules. On the other hand, one of the memory interface ports is dedicated to writing integral image, while the other port is for accessing integral image requested by the mean intensity block. This interface block has a Round-Robin scheduler to arrange the memory access requests, and interface the physical memory.

3.3.1 FAST Corner Detection

FAST corner detection algorithm is based on the intensity comparison between the central candidate pixel P_c and its neighbourhood. It tests 16 locations on the Bresenham circle surrounding P_c. Let a point on this surrounding circle $P_{c\to x}$, be element of S_{bright}, if it has higher intensity than $P_c+\theta$; and be element of S_{dark}, if it has lower intensity than $P_c-\theta$. Here, θ is a predefined threshold. If S_{dark} or S_{bright} has at least 9 contiguous points, then P_c is labelled as a corner.

This comparison is illustrated in Figure 3.18. The pixel in the center with the gray level intensity 10 is under the corner test where threshold θ is 30. Values of the neighbour pixels on the Bresenham circle centered the candidate point is given in the figure. In this case, we find a contiguous arc of 10 pixels which are all brighter than the candidate point more than the threshold $\theta = 30$. Thus, it passes the corner test and marked as a corner.

If a pixel pass the corner test, frequently its adjacent pixels may pass the test as well. However,

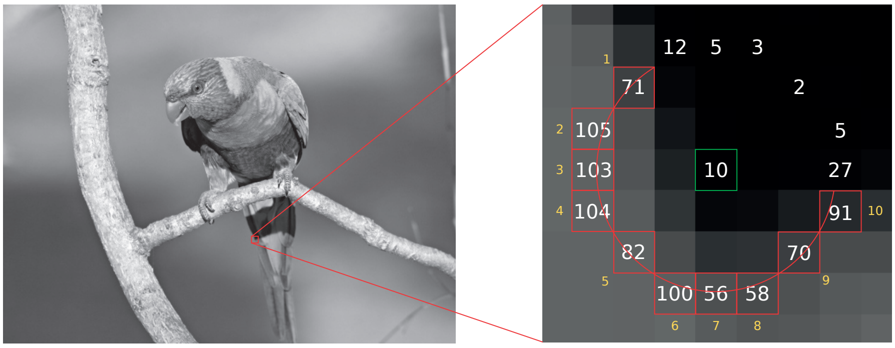


Figure 3.18 – A FAST corner on an example image

the corners next to each other does not carry a distinctive information, and their description results are similar. Therefore, selecting the strongest corner from the adjacent corners and suppressing the others, reduce the number of the corners to be described. To perform this operation, a score function is defined in Equation 3.24,

$$V = max\left(\sum_{x\in S_{bright}} D_x - \theta, \sum_{x\in S_{dark}} D_x - \theta\right) \tag{3.24}$$

where $D_x = |P_{c\rightarrow x} - P_c|$.

To sum up, implementation of the FAST algorithm requires to access the pixels surrounding the candidate point, applying the corner test, computing the corner score and non-max suppression of the corners. In our hardware implementation, received pixel from the pixel bus is directly feed the FAST computation pipeline. At each clock cycle, one result of a pixel is generated, therefore the throughput of the hardware is the same as the camera.

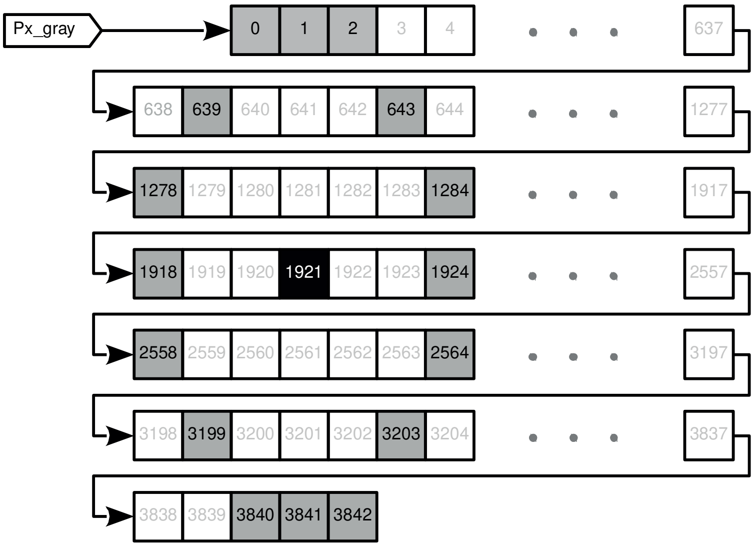


Figure 3.19 – Buffer used in FAST corner detection. Pixel numbers are for VGA resolution.

Buffering the pixel stream enables to access to the neighbouring pixels as shown in Figure 3.19. In this figure, P_c at the center is marked as dark, and its neighbours to be evaluated in the corner test is gray. Thanks to this structures with shift registers, all the pixels for the corner and score calculations will be ready at each clock cycle.

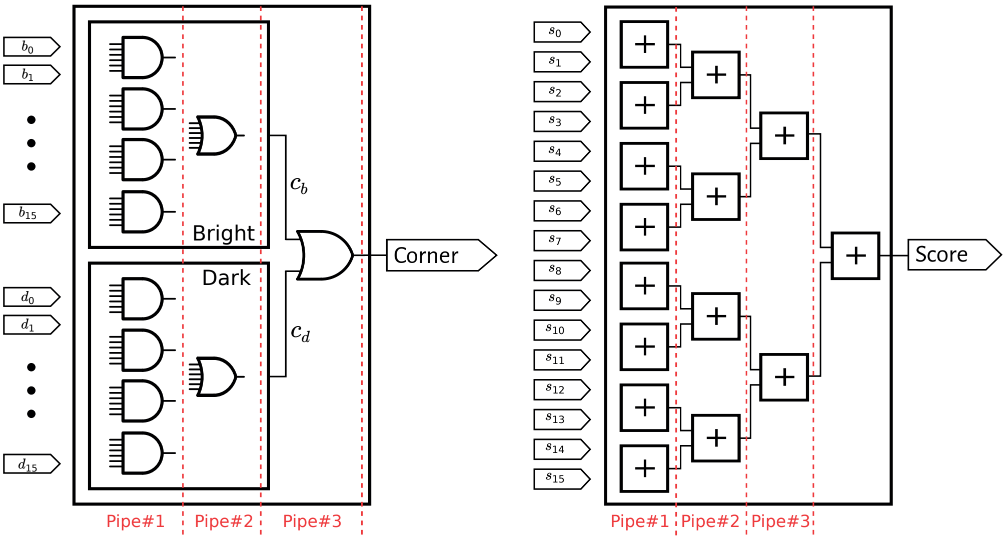


Figure 3.20 – Hardware blocks for FAST corner test and score computation

Hardware blocks of corner test and corner score computation is illustrated in Figure 3.20. In order to reach high frame rates, pipeline registers are used to divide it into stages. The first pipeline stages perform the corner test. Binary inputs to the corner detection block on the left, are the results of the single intensity comparisons:

$$b_i = \begin{cases} 1, & \text{if } P_i > P_c + \theta \\ 0, & \text{otherwise} \end{cases} \qquad d_i = \begin{cases} 1, & \text{if } P_i < P_c - \theta \\ 0, & \text{otherwise} \end{cases} \tag{3.25}$$

Since there are two possible cases darker or brighter arc, this search can be formulated as in the following equation:

$$c_b = \sum_{i=0}^{6} \prod_{j=i}^{i+9} b_i \qquad c_d = \sum_{i=0}^{6} \prod_{j=i}^{i+9} d_i \qquad c = c_b + c_d \tag{3.26}$$

where c is corner test result, c_b and c_d brighter and darker arc search results respectively. This operation can be realized with the logic gates (AND, OR) as depicted.

The corner score is computed by an adder tree in the last stages of the pipeline as the sum of s_i inputs:

$$s = \sum_{i=0}^{15} s_i \tag{3.27}$$

where s_i is determined by brighter or darker arc search:

$$s_i = \begin{cases} |P_i - P_c - \theta|, & \text{if } c_b \text{ and } P_i > P_c + \theta \\ |P_i - P_c + \theta|, & \text{if } c_d \text{ and } P_i < P_c - \theta \\ 0, & \text{otherwise} \end{cases} \tag{3.28}$$

In order to suppress the weaker adjacent corners, accessing the corner scores of the 1-neighbouring pixels is required as given in Equation 3.24. Therefore, similar buffering structure is applied to FAST scores for the non-maximal suppression, as given in Figure 3.18. If only the central point under suppression test is higher than its neighbours, its coordinates is transferred to the next keypoint description block. Otherwise, the corner is suppressed and no description request is produced.

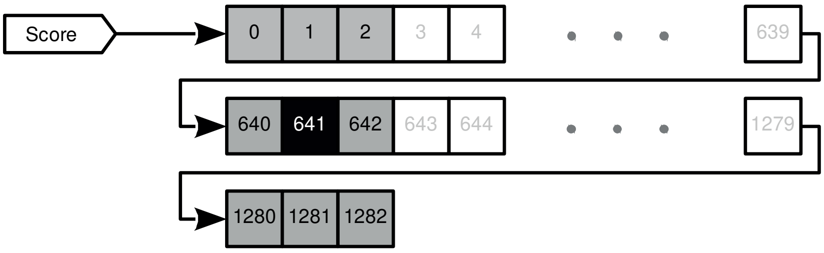
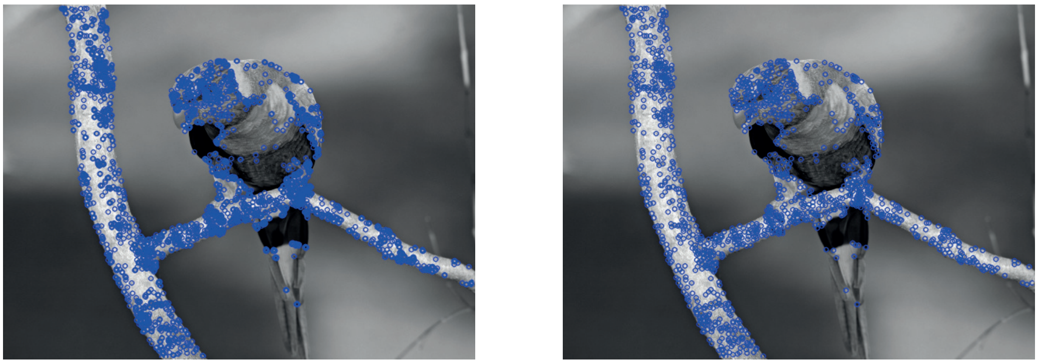


Figure 3.21 – Buffering of scores for non-max suppression



(a) FAST keypoints without NMS

(b) FAST keypoints with NMS

Figure 3.22 – Hardware simulation results of the FAST block

In Figure 3.22b, keypoints found by the hardware simulation are shown. If NMS is disabled, 2915 keypoints are found in Figure 3.22a. When NMS block is activated, number of keypoints are reduced down to 1282 in Figure 3.22b, which corresponds 56% reduction. Hardware implementation of the FAST block produces the same results with OpenCV implementation.

3.3.2 FREAK Description

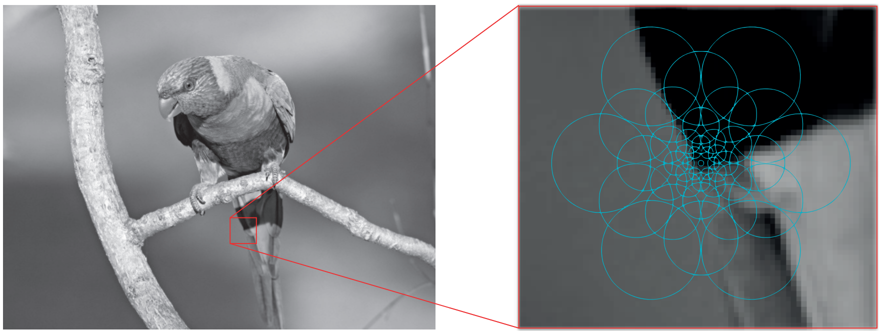


Figure 3.23 – FREAK pattern applied on a corner

In the general Local Binary Description framework, a number of locations are selected with different smoothing radius and comparison between their intensities is performed. The FREAK method consists of a circular overlapping pattern in which the size of Gaussian kernels exponentially increases as depicted in Figure 3.23. This figure also illustrates the description of a keypoint previously detected by FAST corner detection block. There are 7 stages with 6 sampling points (*i.e.*, 42 circles in total). Thus, including the value of the selected keypoint itself, there are 43 values to be compared pair-wise. Among those points, 512 most distinctive comparisons are selected with machine learning methods. For more detailed information, please refer to [64].

Primary operation of the algorithm is the computation of the mean intensities of the Gaussian smoothed circles, and their comparison in the pre-determined order. In its software implementation, instead of generating the Gaussian smoothed intensity regions which is computationally costly, box-average approximation with integral image is used. In our method, we also utilized the box area approach as shown in Figure 3.24. Following subsections, explain these primary sub-blocks, *i.e.*, integral image, mean intensity, and comparison blocks.

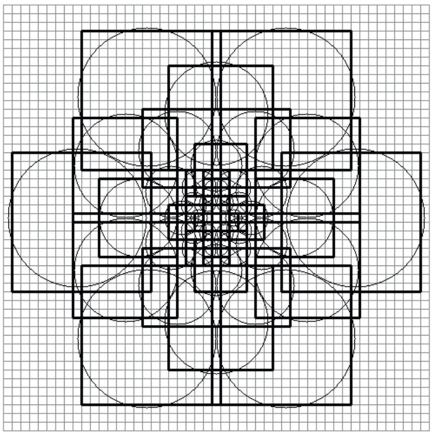


Figure 3.24 – FREAK pattern and box approximation

Integral Image

Integral image is a very efficient tool widely used in feature detection applications [71]. A pixel in the integral image I at (x, y) coordinates is given in Equation 3.29:

$$I(x,y) = \sum_{x' \leq x, y' \leq y} P(x', y'). \tag{3.29}$$

where $P(x', y')$ is the pixel intensity value at (x', y') coordinates of the image. Integral image can be computed during the pixel stream:

$$I(x,y) = P(x,y) + I(x,y-1) + I(x-1,y) - I(x-1,y-1) \tag{3.30}$$

With the integral image method, sum of intensities of a given rectangular area centered at (x, y) with size of r, can be easily found as in the following:

$$I = \frac{I_{UL} + I_{DR} - I_{UR} - I_{DL}}{r} \tag{3.31}$$

where $I_{\mathrm{UL}} = I(x-y, y-r)$, $I_{\mathrm{DR}} = I(x+r, y+r)$, $I_{\mathrm{UR}} = I(x-r, x+r)$, and $I_{\mathrm{DL}} = I(x+r, y-r)$. By using this approximation, both memory bandwidth and computation time is reduced. In order to compute the sum of the pixels inside a rectangle, instead of accessing all the pixels in that area, only four access is required to the integral image regardless of the size of the rectangle.

In this architecture, we benefit from the square-sum approximation like in the software implementation, and utilized integral image as well. Differently from the software approach, integral image can be computed during the pixel flow. In addition, FREAK computations does not necessarily wait for end of the frame. As soon as integral image has enough data in the memory for a keypoint position, computation for that position starts.

Block diagram of the integral image generation hardware is shown in Figure 3.25. A simple hardware structure with a line buffer and an adder, creates the integral image stream to store on the SRAM.

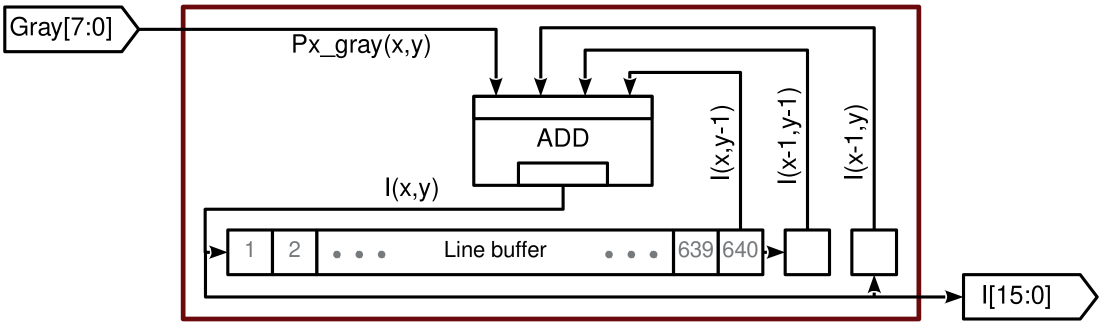


Figure 3.25 – Integral image calculation block

Mean Intensity

Computing the mean intensity of the square-boxes around the keypoint is primary operation for all local binary descriptor implementations. As depicted in Figure 3.17, FREAK module accesses the memory to obtain the integral image values. In order to calculate the mean intensity of a single square box, four values corresponding four memory accesses is required to the integral image according to Equation 3.32. Since the FREAK pattern consists of 43 sampling points, memory access is requested 43 × 4 = 192 times for each keypoint.

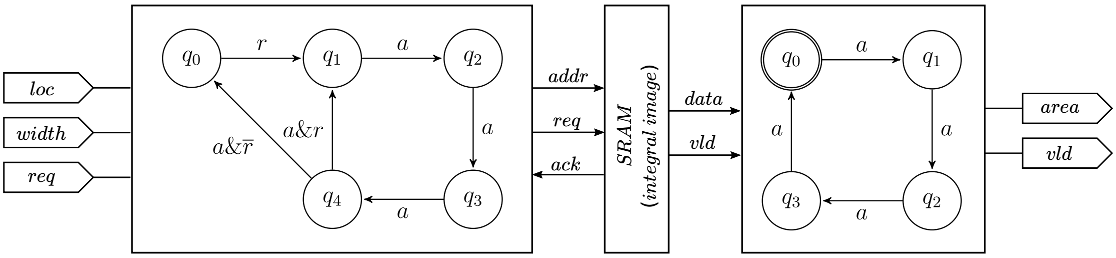


Figure 3.26 – Hardware of the box area computation

In order to manage the access requests to the integral image stored in the SRAM, hardware design consists of two state machines as illustrated in Figure 3.26. The first machine receives the row and column address of the top left corner of the requested square, as well as its width and a request signal. Request signal makes FSM to switch to q_1 state, where it sends the top-left address the square to the memory interface and requests data in the integral image. As it receives the acknowledge signal from the memory interface, it switch to the other states q_2, q_3 and q_4 to request the corner positions of the square. In q_4 state, if there is another area computation request, it goes to q_1 state, and starts the next computations. Otherwise, it goes to q_0 state where it waits the next request in idle mode.

The second FSM which receives the integral image data I_{UL}, I_{UR}, I_{DR}, I_{DL} respectively. For each data with acknowledge signal, FSM changes its state and add or subtract the integral data

r=	2	3	4	6	9	12	16
s_1	$A \ll 8$	$A \ll 7$	$A \ll 6$	$A \ll 4$	$A \ll 3$	$A \ll 2$	$A \ll 2$
s_2	0	$-A \ll 4$	0	$A \ll 3$	$A \ll 2$	$A \ll 1$	0
s_3	0	$\mathrm{A} \ll 1$	0	$\mathrm{A} \ll 2$	A	A	0
$s \gg 10$	0.25	0.1113	0.0625	0.0273	0.0127	0.0069	0.0039
$A \div r^2$	0.25	0.1111	0.0625	0.0278	0.0123	0.0068	0.0039
error	0 %	0.2%	0%	1.8%	3.3%	1.4%	0%

Table 3.1 – LUT approximation of $A \div r^2$ for FREAK pattern

from the area register, depending on the sign in the Equation 3.32:

$$A = I_{UL} + I_{DR} - I_{UR} - I_{DL} \tag{3.32}$$

In the last stage, sum of the intensities must be divided by the area of the square. FREAK pattern has 7 levels and each level has identical circles. Therefore, in square box approximation, there are 7 different square dimensions which limits the set of dividers to 7. In this case, instead of using divider-multiplier to compute A/r^2 term we can use only low-cost multiplexer and adder, benefiting from the zero-cost logic shift operations. This operation can be formulated as sum of three terms as follows:

$$s = s_1 + s_2 + s_3 \qquad\qquad m = s \gg 10 \tag{3.33}$$

Although this approximation brings a maximum 3.3% difference between the results from the floating division operation, it is still acceptable due to its reduction on the hardware resource cost.

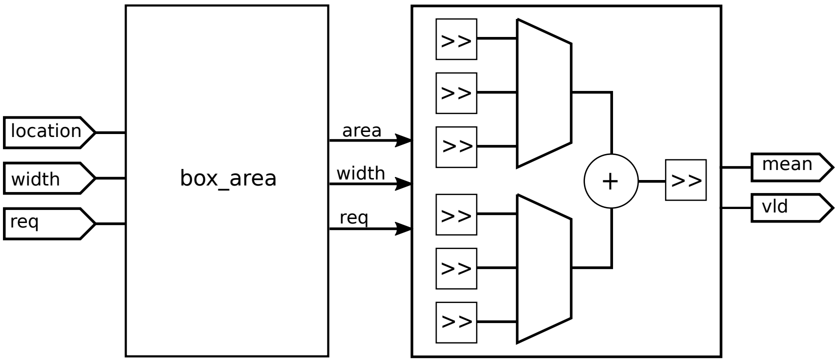


Figure 3.27 – Hardware of the box area computation

Relative corner positions are calculated and embedded into a look-up-table (LUT) before the synthesis. When a keypoint location arrives at the block with a data valid signal, the computation is triggered as soon as integral image has enough data for the boxes of FREAK pattern. SRAM address is computed by the sum of the keypoint location input and these relative corner positions.

Comparison

Another critical block of the FREAK algorithm is comparison of those 43 smoothed intensity points calculated by the mean intensity block with respect to a specific pattern. In the original software implementation, a pre-selected comparison pattern is applied with 512 pairs to construct the 512-bit binary descriptor. Intensity comparison is performed as given by:

$$T(P_a) = \begin{cases} 1, & \text{if}\big(I(P_a^{r_1}) - P_a^{r_2}\big) > 0, \\ 0, & \text{otherwise.} \end{cases} \tag{3.34}$$

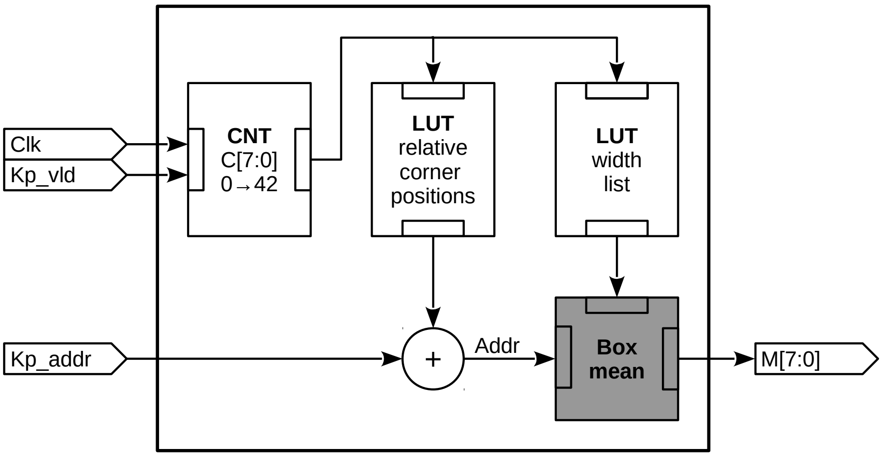


Figure 3.28 – Instantiating the box mean intensity computation block for FREAK pattern

The comparisons then build the binary descriptor:

$$D = \sum_{0 \le a < N} 2^a T(P_a) \tag{3.35}$$

FREAK description result consists of comparisons of the selected intensity values. Therefore, the output of the mean intensity block directly feeds the comparison block. Block diagram of the comparison unit is shown in Figure 3.29. 512 most distinctive comparisons were selected instead of comparing all the pairs in the software implementation. The comparison pair index are pre-calculated and stored in a LUT. Received intensity values from the mean intensity block are multiplexed and stored in the specific registers according to this LUT. When the mean intensity block completes all the calculations, this block performs the comparison in parallel following the arrival of all the mean intensities of the keypoint. Finally keypoint coordinates and their calculated 512-bit FREAK descriptions are given to the 8-bit output in a sequential order.

Metadata sequencer

In this hardware, metadata bus designed as 8-bit, therefore the outputs of FAST and FREAK blocks must be formed into 8-bit packages. In order to produce this stream, a sequencer circuit is needed. We designed a simple Finite State Machine (FSM) with 4 states as can be seen in Figure 3.30. In the first state, it waits for a request signal from the FAST corner detection module. If a pixel passes the corner test, state machine switches to keypoint transfer (KP tx) state and transmits the row and column address of the corner. For VGA resolution, row and column address needs to be stored in 10 bits each, therefore coordinates are transferred in 4 of 8-bit packages.

After FREAK block completes its operations, it generates 512-bit description result to be transferred. In this case, FSM is triggered by `freak_req` signal and changes state from FREAK

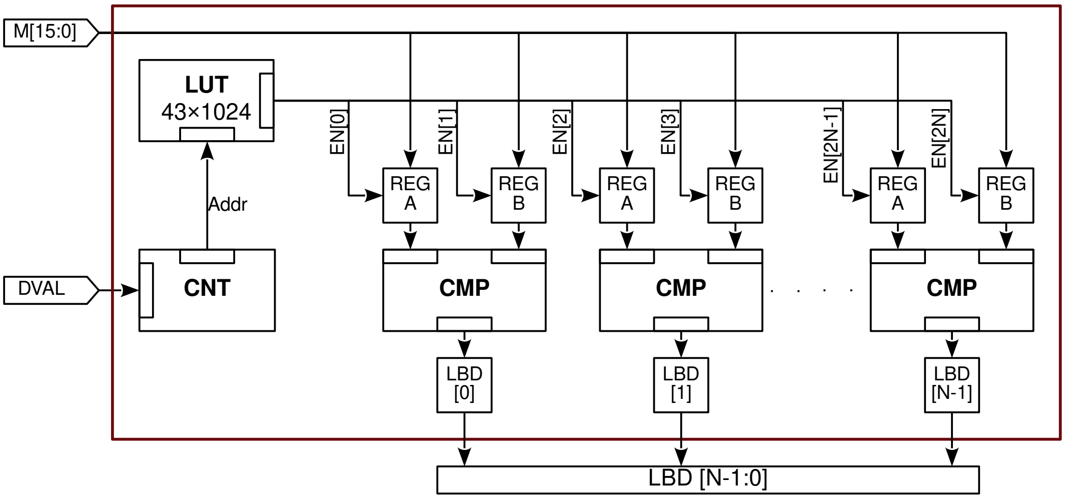


Figure 3.29 – Parallel comparison of mean intensities

wait to FREAK tx where it transmits the description result in 64 clock cycle by help of the counter `cnt2`. When the transfer is completed, FSM goes into KP wait state and observes a request from the FAST block.

3.4 Hybrid Processor Population

Operation speed of the traditional computation methods based on consecutive operations are limited by the clock frequency of the processing unit. Although very high frequencies can be reached today, biological neural networks show much superior performance in terms of timing and power efficiency. It is known that human brain consumes 20 W [72], although a supercomputer requires megawatts of power [73] for similar tasks. Therefore, this efficiency has attracted researchers' attention, and developing bio-inspired computation devices has been a challenge among them.

One of the earliest attempts to the computation methods with bio-inspired structures is Cellular Neural Networks(CNN) [74] [75]. In the architectural point of view, a standard CNN is a system of locally coupled non-linear processing units placed on a rectangular grid as shown in Figure 3.31. As it is illustrated in this figure, the cell at the position i, j is connected to the cells only within its neighbourhood $N_r(i, j)$. Because of this local connections, data can be shared with only the neighbors, but propagation enables temporal interactions. Dynamics of the non-linear cells is commonly given in the following state equation:

$$\dot{x_{i,j}}(t) = \sum_{k,l \in N_r(i,j)} A_{k,l} x_{k,l}(t) + \sum_{k,l \in N_r(i,j)} B_{k,l} u_{k,l}(t) + I_{i,j} \tag{3.36}$$

where x is output of the cells, A is output feedback operator, B is input control operator, and I

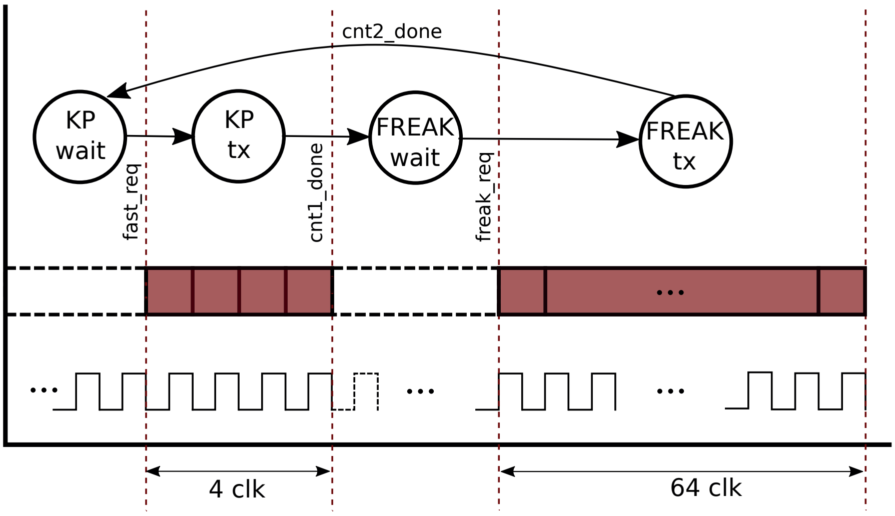


Figure 3.30 – FSM and timing diagram of metadata sequencer

is the input of the cell.

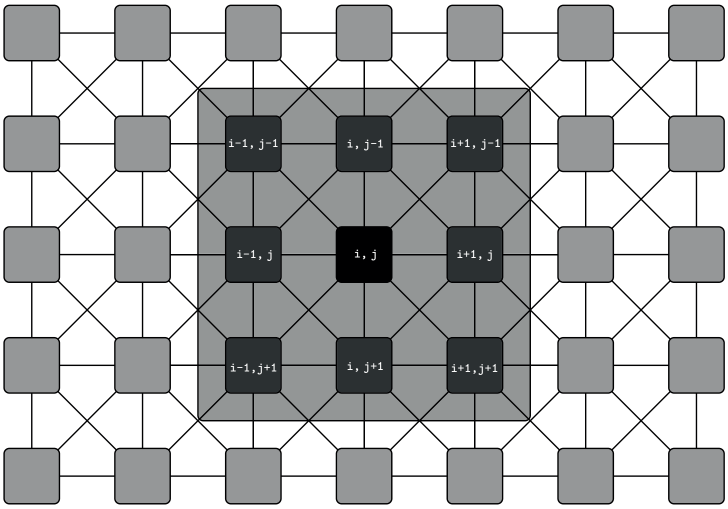


Figure 3.31 – Cell connections in CNN

The first analog circuit design of the CNN is presented in [76] with the name of CNN Universal Machine (CNN-UM). It is promoted as the first algorithmically programmable analog array computer and the 'analogic' (analog+logic) algorithms of this new computing method is explained. A successful analog processing array of 128 × 128 cells is implemented in a $0.35\mu m$ technology [77]. In that work, the processor array is used in real-time image processing applications and have embedded distributed optical sensors. It achieves to process VGA frames at 100 fps by applying some basic image processing tasks.

Although CNN is designed initially as analog, there are many digital emulators implemented on FPGA and ASIC. Continuous time domain expressions in Equation 3.36 can be discretizate with Forward Euler method:

$$x_{i,j}(n+1) = x_{i,j}(n) + h\left(\sum_{k,l\in N_r(i,j)} A_{k,l}x_{k,l}(n) + \sum_{k,l\in N_r(i,j)} B_{k,l}u_{k,l}(n) + I_{i,j}\right) \tag{3.37}$$

where h is the time step. A digital implementation of the CNN-UM is presented in [78]. In that work, authors implemented a multi-layer CNN model on a Virtex-300 FPGA.

The biological networks such as in frontal cortex performs their operations by benefiting form inhibition and excitation activities. A mathematical model of inhibitory and excitatory sub-populations constituting of the network is given by Wilson and Cowan [79]. Their model characterize the dynamics of the neural population explicitly with $E(t)$ and $I(t)$ variables, being proportion of excitatory and inhibitory cells firing per unit time at the instant t, respectively. Based on this network model, Ayhan and Yalcin used a similar network to realize an artificial olfaction system [80] and implemented in FPGA [81]. They generate different networks by randomly distributing the inhibitory and excitatory cells, and select the best random distribution by observing their classification performances. In the implementation point of view, this method does not require a significant extra complexity over classical CNN. It needs an identity matrix with the size of the network.

In this section, we develop a Hybrid Processor Population model and give its FPGA implementation for vision applications. We have integrated this hardware block into our smart camera architecture. Since the cell types can be configured by identity matrix, the network is highly configurable. We presented different network configurations, directly can be used in local binary features applications as parallel subtractors. Moreover, this approach can lead different methods for feature extraction.

3.4.1 Cell Model

The network architecture is based on the Cellular Neural Network which is composed of locally coupled cells (or Neural Processing Elements, NPEs) on a two-dimensional grid of size $M \times N$. Similar to [81], each cell can be two different types of characteristic as excitatory and inhibitory. Cell types are stored in H matrix of size $M \times N$, and elements of the matrix $h_{i,j} \in \{0,1\}$ indicate the cell type as inhibitory (0) and excitatory (1). Since the implementation target is digital hardware realization, cell dynamic can be written directly in the discrete time domain:

$$x_{i,j}[n+1] = c_0 x_{i,j}[n] + \sum_{k,l\in S} c_1.y_{k,l}[n] + c_2.g_{in}[n] \tag{3.38}$$

where i, j, k, l are indices represent the location of the NPE on the grid network:

$$1 \le i,k \le M; 1 \le j,l \le N \tag{3.39}$$

In this equation, next value of the state variable $x[n+1]$ is a linear combination of the current value $x[n]$, input (g_{in}), and the output values $(y_{k,l})$ of the neighbouring cells (S). If the state variable x exceeds the limits (X_{max}, X_{min}), output y saturates. Output of a cell is defined as in the following non-linear equation:

$$y_{i,j}[n] = \begin{cases} X_{max}, & x_{i,j}[n] > X_{max} \\ X_{min}, & x_{i,j}[n] < X_{min} \\ x_{i,j}[n], & \text{else} \end{cases} \tag{3.40}$$

The binary outputs constitute the binary descriptor, while y output is the input of neighbour cells. If a threshold θ is applied to state variable x, binary output is produced:

$$b_{i,j}[n] = \begin{cases} 0, & x < \theta \\ 1, & \text{else} \end{cases} \tag{3.41}$$

In Equation 3.38, c_1 is a function of identity matrix H and determines the weight of the connections between the cell-types. If h is concatenation of the identities of the neighbouring cells $(h = [H(i,j), H(k,l)])$, different c_1 connection weights can be defined as:

$$c_1 = c_1(H; i,j; k,l) = \begin{cases} c_1^{II}, & h = 00 \\ c_1^{IE}, & h = 01 \\ c_1^{EI}, & h = 10 \\ c_1^{EE}, & h = 11 \end{cases} \tag{3.42}$$

In this equation, I and E superscripts represent inhibitory and excitatory identity, respectively. Connection weights from inhibitory cells are defined as negative $(i.e., c_1^{IE}, c_1^{II} \leq 0)$. In fact, the network consists of identical cells like in the CNN approach, however identity input of the cells define their behaviour as either inhibitory or excitatory.

3.4.2 Connectivity Rules

In the classical CNN structure, a single cell may have connected with both positive and negative weights to its neighbours. In contrast, visual cortex is compound of the neurons which show either inhibitory or excitatory behaviour. Limiting the connections create isolated regions which are not affected by the other cells, and this extends the flexibility of the network. In order to provide this, connections between the same type of cells can be restricted as in biological examples:

$$c_1^{EE} = c_1^{II} = 0 \tag{3.43}$$

The identity matrix is a representation and each element of the matrix is directly connected to cell. This can be implemented by the help of an XOR operation applied on the identities:

$$c_{i,j} = h_{i,j} \oplus h_{k,l} \tag{3.44}$$

We can define another connection limitation rule by prohibiting the diagonal neighbourhood with d parameter. In case of $d = 0$, each cell has 4 connections (left, right, up and down), otherwise ($d = 1$) 8 connections including diagonal neighbours. This helps to create isolated sub-networks. As a result, if all cell types are connected and diagonal neighbourhood is permitted, connectivity of the network is maximum. On the other hand, connections are limited by synaptic rules and this increase the flexibility of the architecture. Therefore a cell can be connected from 0 up to 8 neighbouring cells depending on the rules and type of surrounding cells.

3.4.3 Example Network Configurations

Due to the flexibility of the architecture, many different types of network can be generated. For instance if the inhibitory and excitatory cells are distributed into the network in a particular order, the network can be used as an accelerator in LBP and LBD operations. In addition, if inhibitory and excitatory cells are distributed in a non-uniform order under some connectivity rules, cells can have different number of connections from each other. Moreover, a longer process of the network enables a more complex calculation with propagation effect. By using random distributions and propagation, novel binary descriptors can be produced.

Besides being accelerator/co-processor of the local binary description process, reconfiguration is a key-concept in this architecture. Since the networks are described by identity matrix and connectivity rules, a modification on those parameters corresponds a new network. Therefore, in case of different binary description operation is a requirement for the multiple target applications, this network provide to this ability without any additional hardware cost. In this sub-section, 3 different application examples will be examined.

Local Binary Descriptor (LBD)

In this example, selected interest points are described by LBDs. Gaussian smoothed intensity points are selected and compared around an interest point. 512 intensity comparison is required to obtain 512-bit binary descriptor. Therefore, parallelizing this computation results in accelerating the process. For this application, the connectivity rules should be selected as in the following:

$$c_1^{EE} = c_1^{II} = c_1^{EI} = 0, c_1^{IE} = -1, d = 0 \tag{3.45}$$

Identity matrix and corresponding network model is depicted in Figure 3.32. As a result of imposing this connectivity rules and identity matrix on the network, isolated subtraction couples are generated. For each set, the network is operated one cycle, excitatory cells take value of difference of two intensity points. A threshold $\theta = 0$ is applied to the state variable in order to obtain the binary output. By this way, the network can be used as a LBD co-processor which accelerates the calculation by parallelizing $M \times N \div 2$ subtraction operations.

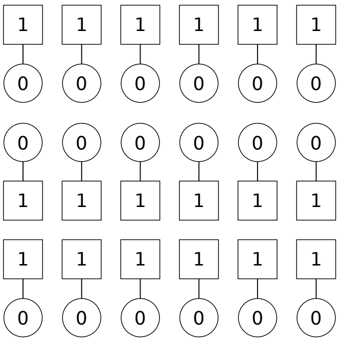


Figure 3.32 – A network configuration for LBDs

Local Binary Pattern (LBP)

Local Binary Pattern (LBP) is a very efficient texture operator and it has especially superior performance in face recognition applications [82]. In this operation, pixel value of a keypoint is subtracted from its neighbours, and a threshold is applied to remaining values. The binary result represents the LBP of the corresponding pixel. To speed up the process, this subtract and threshold operation can be parellelized by using the proposed keypoint detection hardware. The network connectivity parameters should be set as follows:

$$c_1^{EE} = c_1^{II} = c_1^{EI} = 0, c_1^{IE} = -1, d = 1 \tag{3.46}$$

A proper identity matrix creates $M \times N \div 9$ isolated groups with dimension of 3×3 as depicted in Figure 3.33. In this network, inhibitory cells are surrounded by excitatory cells and the inputs of those cells are the image pixel values. Similar to LBD example, a threshold $\theta = 0$ is applied to the state variable x, thus the binary output b is generated. As a consequence, excitatory cell output gives the desired LBP values following a bit ordering manipulation.

Non-Uniform Distribution

Inhibitory-excitatory cells are distributed in a particular order to create isolated sub-networks for LBD and LBP operations. Conversely to these applications, non-uniform or random connections can be seen in biological networks [83]. If inhibitory and excitatory cells are distributed in a non-uniform order, connections give different combinations because of the

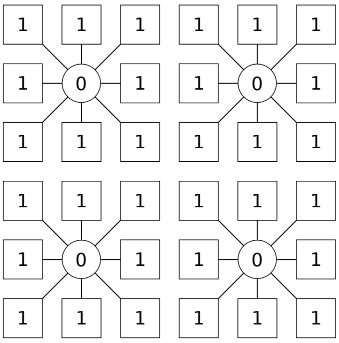


Figure 3.33 – A network configuration for LBPs

connectivity rules. As it can be seen as highlighted in Figure 3.34, cells with different number of connections (1, 2, 3 and 4) are available in the network. Therefore differently from current LBDs, more than two intensity values can be considered for one bit of the descriptor. This can increase the accuracy of the description, since a selected combination (*e.g.* $I_1 - I_2 - I_3$ or $I_1 - (I_2 - I_3)/2$) may represent more distinctive information than a $I_1 - I_2$ comparison selected by FREAK in particular applications, where I_n represent an intensity value of a smoothed sampling point.

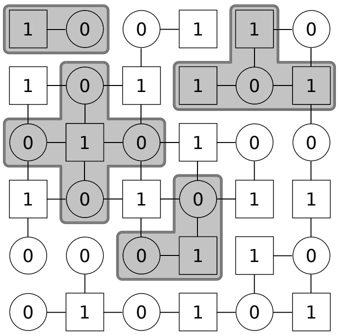


Figure 3.34 – Randomly distributed inhibitory and excitatory cells under limited connectivity rules.

3.4.4 Realization

In the hardware implementation of a single HPP cell, all connection weights ($c_1^{II}, c_1^{IE}, c_1^{EI}, c_1^{EE}$) are limited with to be either 0 or 2^{-N} ($N \in \mathbb{N}$). Thus, $c_1.y_n$ terms can be calculated by a zero-cost casting operation. Consequently, only low-area hardware units like adder and multiplexer take place in the hardware which is depicted in Figure 3.35.

Building the HPP network needs a large multiplexer circuitry to map the input registers to the HPP cells as shown in Figure 3.36. Depending on the cell identities (h) and connectivity rules, some of the connections between the cells are set to zero automatically. Contributions of the neighbouring cells are calculated in $c_1.y_n$ units by setting the cast operations before

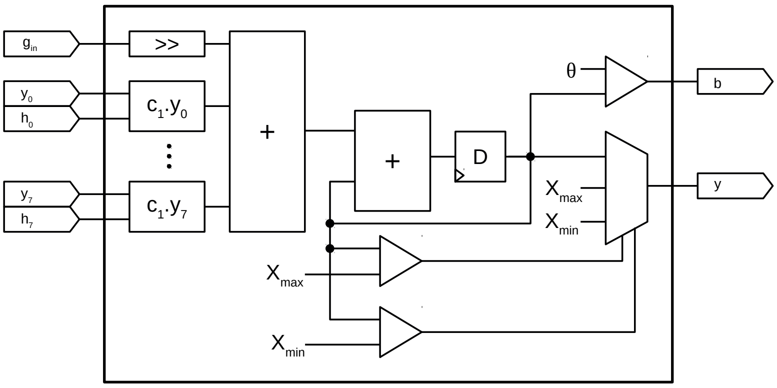


Figure 3.35 – Hardware of a single cell in HPP

synthesis. On the other hand, cell identities and connectivity rules can be changed in run-time by a control unit.

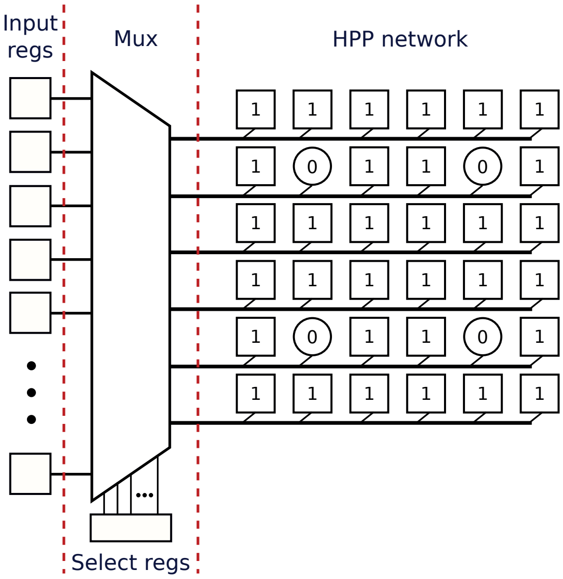


Figure 3.36 – Multiplexing the inputs for HPP

3.4.5 Conclusion

In this section, we extended our smart camera architecture which consists of keypoint detection and description blocks, with Hybrid Processor Population (HPP). HPP is designed as a highly configurable processing block. By changing identity matrix and synaptic rules, many network configurations can be generated. Among those networks, two different configurations are shown to be incorporated with keypoint detection and description operations. In addition to this, the HPP network is also capable of performing complex comparisons and multiple iterations.

3.5 Result

In this chapter, we presented our smart camera architecture which consists of three processing unit: moving object detection, keypoint description and cellular neural network. For their implementations, firstly we coded their models in Python language.

OpenCV is an open source computer vision library which can be imported into the Python. Since MOG background subtraction, FAST keypoint detection and FREAK description methods are already implemented in OpenCV library, accuracy of the proposed algorithms can be evaluated by comparing them. In this stage, we also created test vectors and their expected results to be used in the hardware simulations.

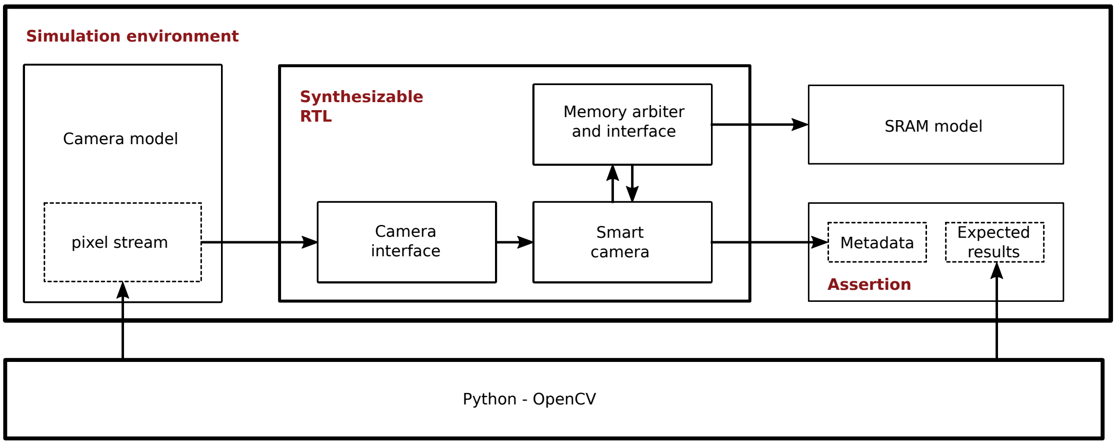


Figure 3.37 – Hardware simulation environment of smart camera

Then, we described all these hardware processing blocks in VHDL. In addition to this, behavioral models of the camera (Toshiba TCM8230 VGA) and the memory (SRAM) and their interfaces are also coded in VHDL in order to create a board level cycle-accurate simulation environment as depicted in Figure 3.37. Including these VHDL codes and applying the test vectors, we verify the functionality with behavioral simulations in ModelSim.

Following this, we synthesized the design for VGA resolution with Vivado, targeting Xilinx VC707 board with Virtex-7 XC7VX485T FPGA, however since there is no IP core from a vendor, full design can easily be adapted to other technologies. As an example, some functions are taped out for 65 nm chip technology in Section 4.4.

We set the system clock frequency to 100 MHz and frame rate to 30 fps, then we run post-implementation simulation to extract switching activities of the logic blocks, and used Xilinx Power Estimation Tool with this information. According to its output, power consumption of processing of a VGA frame with 1200 keypoints is found as 292 mW for feature extraction module. In fact, since device static power is 242 mW, dynamic logic consumes 50 mW. On the other hand, for moving object detection block, dynamic power consumption is estimated as 18 mW (*i.e.* 260 mW with device static).

On-the-fly pixel processing of the blocks needs a number of line buffers, a form of shift registers. Shift registers are mapped onto slice LUTs benefiting from SRL32 structure of Xilinx FPGA technology. This mapping brings an area efficiency. Hardware resource allocation of the processing blocks is given in Table 3.2. Since there is no multiplier in any blocks, DSP blocks remains free. Block RAM is not also utilized, but in case of internal storage of the integral image in feature extraction block, it will be occupied by 56%.

Table 3.2 – Resource allocation

Blocks	Slice LUTs	Slice Registers	DSP	BRAM
FAST	2541 (0.84%)	1183 (0.19%)	0	0
FREAK	2356 (0.78%)	531 (0.09%)	0	0
HPP	2952 (0.97%)	1836 (0.01%)	0	0
Total	8941 (3%)	4653 (0.77%)	0	0

According to these simulation analysis results, a performance comparison for Moving Object Detection is given in Table 3.3, and for Feature Description in Table 3.4. Considering these computation blocks separately, we can prove that both modules outperform the other designs in terms of power, area and speed tradeoff.

Table 3.3 – Performance comparison for Moving Object Detection

Work	Max. Frame Rate	Power (mW)	Area [LUT+REG]
[28]	15	670	0.8 k
[42]	91 (HD)	136	1 k
[45]	182	459	0.8 k
This work (MOD)	150	230	0.9k

Table 3.4 – Performance comparison for Keypoint Description

Work	Max. Frame Rate	Power (mW)	Area [LUT+REG]
[84]	55	N/A	31 k
[69]	325	2400	16 k
This work (FD)	300	580	7 k

To conclude, processing blocks which constitutes the smart camera are implemented in a hardware efficient way. Having processing blocks at different energy levels make the system flexible for power-critical applications. Since feature extraction is more expensive in terms of power, its operation can be decided if there is a moving object detected by moving object detection block, also taking into account of energy level of the system.

4 Applications

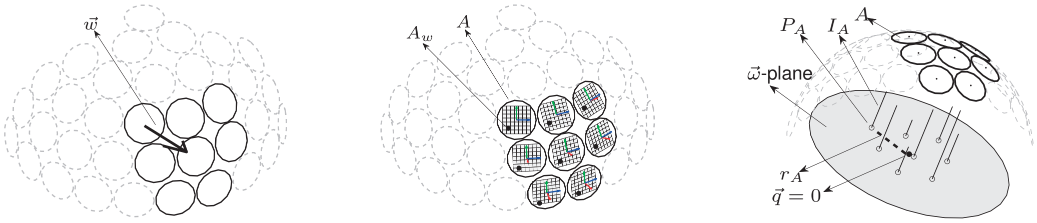
4.1 Omnidirectional Image Reconstruction

The number of cameras determine the full-coverage distance of the scene, thus different solutions could be produced according to the application by changing the number and distribution of the cameras. In order to cover short distances, it is required to increase the number of cameras. However, the computational complexity and memory access load increases proportional to captured images. Moreover, computer vision applications such as feature detection and description, object recognition and tracking increase the random access storage and bandwidth requirements. In order to satisfy the requirements, speed- and memory-optimized approaches to algorithms as well as the amelioration of the computational performance of the embedded devices in terms of memory, power, and processing unit capacity should be considered.

In this part, we propose a hybrid architecture examining centralized and distributed approaches. This architecture distribute the computational load into processing nodes which is connected with a network with tree topology. In addition, we propose a hardware solution for processing nodes of the network, which targets scalability and maximizing memory storage and bandwidth capabilities. For this purpose, firstly omnidirectional image reconstruction algorithm will be given. Following, centralized and distributed approaches to algorithm implementation will be explained. Finally, a novel hybrid architecture and its hardware solution with the smart camera architecture will be presented.

4.1.1 Method

The omnidirectional vision of a virtual observer located anywhere inside the hemisphere of the Panoptic structure can be reconstructed by combining the information collected by each camera in the light field [85]. In this process, the omnidirectional view is estimated on a discretized spherical surface S_d of directions. The surface of this sphere is discretized into an equiangular grid with N_θ latitudes and N_ϕ longitudes samples, where each sample represents



(a) (b) (c)

Figure 4.1 – (a)Cameras contributing to the direction $\vec{\omega}$, (b) contributing pixel positions on the image frame of the contributing cameras for direction $\vec{\omega}$, (c) projections of camera centers contributing to direction $\vec{\omega}$ onto planar surface normal to $\vec{\omega}$.

one pixel. Different pixel distributions over the sphere are possible and a comparison can be found in [17]. A unit vector $\vec{\omega} \in S_d$, represented in the spherical coordinate system $\omega = (\theta_\omega, \phi_\omega)$, is assigned to the position of each pixel.

The construction of the virtual omnidirectional view $\mathscr{L}(\vec{q}, \vec{\omega}) \in R$, where $\vec{q}$ determines the location of the observer, is performed in two steps. The first step consists of finding a pixel in each camera image frame that corresponds to the direction defined by $\vec{\omega}$. Due to the rectangular sampling grid of the cameras, the $\vec{\omega}$ does not coincide with the exact pixel grid locations on the camera image frames. The pixel location is chosen using the nearest neighbour method, where the pixel closest to the desired direction is chosen as an estimate of the light ray intensity. The process is then repeated for all $\vec{\omega}$ and results in the estimated values $\mathscr{L}(c_i, \vec{\omega})$, where c_i is the radial vector directing to the center position of the i^{th} contributing camera's circular face. Figure 4.1a shows an example of the contributing cameras for a random pixel direction $\vec{\omega}$. The contributing position A_ω of the camera A, providing $\mathscr{L}(c_A, \vec{\omega})$ is also indicated in Figure 4.1b. The second reconstruction step is performed in the space of light rays given by direction $\vec{\omega}$ and passing through the camera center positions. Under the assumption of Constant Light Flux (CLF), the light intensity remains constant on the trajectory of any light ray. Following the CLF assumption, the light ray intensity for a given direction $\vec{\omega}$ only varies in its respective orthographic plane. The orthographic plane is a plane normal to $\vec{\omega}$. Such plane is indicated as the "$\vec{\omega}$-plane" in Figure 4.1c, and represented as a gray-shaded circle. The light ray in direction $\vec{\omega}$ recorded by each contributing camera intersects the $\vec{\omega}$-plane in points that are the projections of the cameras' focal points on this plane. The projected focal points of the contributing cameras in $\vec{\omega}$ direction onto the $\vec{\omega}$-plane are highlighted by hollow points in Figure 4.1c. Each projected camera point P_{c_i} on the planar surface is assigned the intensity value $\mathscr{L}(c_i, \vec{\omega})$, that is calculated in the first step.

When applying the nearest neighbour (NN) technique in the second reconstruction step, the light intensity at the virtual observer point for each $\vec{\omega}$ direction is set to the light intensity value of the best observing camera for that direction. The nearest neighbour technique as in the following:

$$
\begin{aligned}
dj &= \operatorname{argmin}_{i \in I}(r_i) \\
L(\vec{q},\vec{\omega}) &= L(c_j,\vec{\omega})
\end{aligned}
\tag{4.1}
$$

where $I = \{i|\vec{\omega}\cdot\vec{t}_i \geq \cos(\frac{\alpha_i}{2})\}$ is the index of the subset of contributing cameras for the pixel direction $\vec{\omega}$. A pixel direction $\vec{\omega}$ is assumed observable by the camera c_i, if the angle between its focal vector $\vec{t}_i$ and the pixel direction $\vec{\omega}$ is smaller than half of the minimum angle of view α_i of camera c_i. The length r_i identifies the distance between the projected focal point of camera c_i and the projected virtual observer point on the $\vec{\omega}$-plane. The camera with the smallest r distance to the virtual observer projected point on the $\vec{\omega}$-plane is considered the best observing camera. As an illustration, such distance is identified with r_A and depicted by a dashed line for the contributing camera A in Figure 4.1c.

Different brightness levels between cameras and misalignment causes sharp transition among the cameras. In order to resolve this transition problems, several other blending techniques were proposed. For example, the linear blending scheme incorporates all the cameras contributing into a selected $\vec{\omega}$ direction through a linear combination [15]. This is conducted by aggregating the weighted intensities of the contributing cameras. The weight of a contributing camera is the reciprocal of the distance between its projected focal point and the projected virtual observer point on the $\vec{\omega}$-plane, r_A in Figure 4.1c. The weights are also normalized to the sum of the inverse of all the contributing cameras distances.

The linear blending is expressed in Equation 4.2.

$$
\begin{aligned}
L(\vec{q},\vec{\omega}) &= \frac{\sum\limits_{i\in I} w_i \cdot L(c_i,\vec{\omega})}{\sum\limits_{i\in I} w_i} \\
w_i &= \frac{1}{r_i}
\end{aligned}
\tag{4.2}
$$

For detailed discussion on camera arrangement on spherical surface, different blending approaches, camera orientation and multiple camera calibration, the reader is referred to [16].

4.1.2 Central Approach

The first implementation approach to omnidirectional reconstruction algorithm is the centralized approach, where data acquisition and data processing reside on the same unit. The architecture of the central approach is depicted in Figure 4.2. The arrow lines shows the flow of image data inside the FPGA. Image data streaming from the cameras enters the FPGA via the camera connectors. A time-multiplexing mechanism must be conducted to store the incoming frame data from all the camera modules onto one of the single data port SRAMs. Hence the memory arbiter block time multiplexes the data received by the camera input

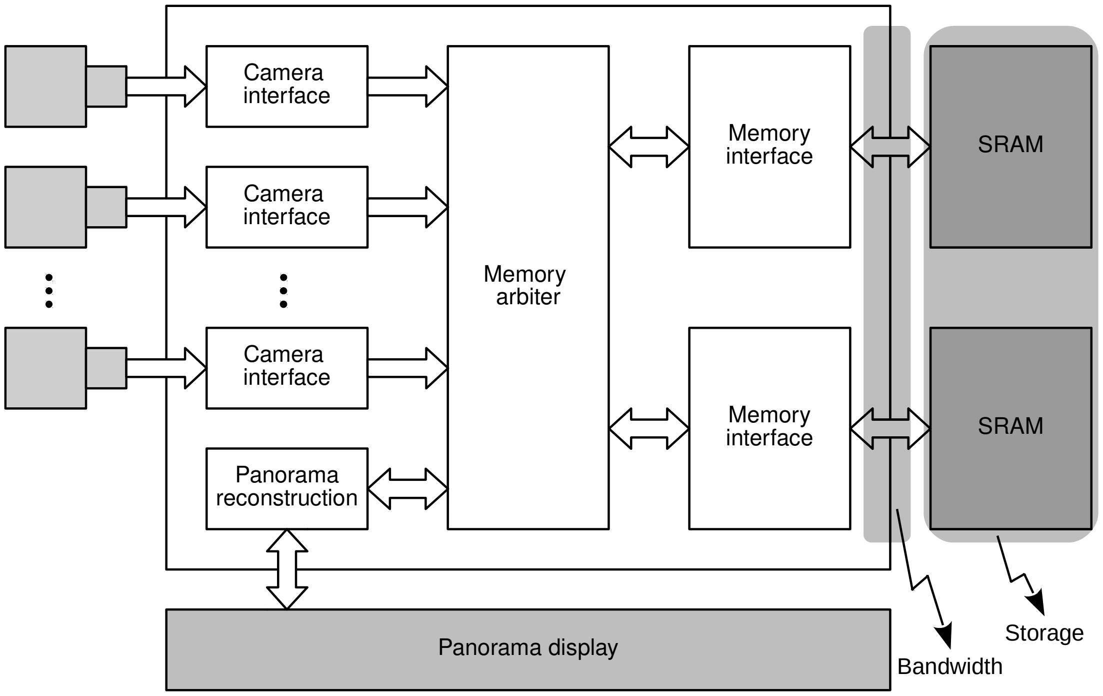


Figure 4.2 – Centralized implementation approach to the reconstruction algorithm. Camera data is saved onto one of the SRAMs, via camera interface and memory interface. Panorama reconstruction algorithm access the camera data which belongs to previous frame.

channel block and transfers it to the memory controller block for storage in one of the SRAMs. The memory controller block interfaces with two external SRAMs available on the board and provides access for storing/retrieving the incoming/previous twenty image frame data on the SRAMs. The SRAMs swap their role (one being written into, and one being read) with the arrival of each new image frame from the cameras. The application unit block is in charge of signal processing and basic functionalities such as single/dual video channel streaming, all channel capture and omnidirectional vision construction. This block accesses the SRAMs via the memory controller block and transfers the processed/image data to the control unit block. The control unit block provides transmission capability over external interfaces available on the board such as high speed LVDS serial links or USB 2.0. The latter block also applies the control commands originating from the computer/FPGA.

A custom FPGA board has been designed using a Xilinx Virtex-5 FPGA as a core processing unit in order to capture and process the video streams produced by the cameras in real-time. The devised system consists of four layers: 1) image sensors, 2) FPGA boards handling local image processing, 3) one central FPGA board for control, external access and last stage image processing, 4) a PC in charge of the applicative layer consisting of displaying the operation results transmitted from the central FPGA board. This board interfaces with twenty single-chip Common Intermediate Format (CIF, 352 × 288) cameras. It contains two Zero Bus Turn around (ZBT) Static Random Access Memories (SRAM) with 36 Mb capacity and an operating bandwidth of 167 MHz, for each. The maximum achievable throughput using this SRAM is approximately 3 Gbps. The details of the implementation is explained detailed in [16].

In the centralized approach, because of the input-output (I/O) constraints, the number of cameras that can be connected to a single node is limited. Furthermore, for high number of cameras and high camera resolutions, the memory bandwidth requirement increases significantly. Therefore, centralized approaches for such systems may create bottlenecks and limit the scalability.

4.1.3 Distributed Approach

The real-time implementation of multi-camera systems applications with a high number of cameras, high image sensor resolutions and the current image sensor architectures demands a high amount of hardware resources, and depending on the target application, it might also demand high computing performances. Parallel processing approaches aim to overcome the limitations of centralized approach by distributing signal processing tasks and memory bandwidth usage among several signal processing blocks. This technique creates the possibility of constructing higher resolution images beyond centralized approach. Moreover, parallel approaches are faster implementations compared to the centralized approaches, which create the possibility of creating higher resolution images. A distributed and parallel implementation is presented in [17].

4.1.4 Distributed and Parallel Implementation

If tasks are distributed properly among many processors, the computation time will decrease significantly. In order to distribute the tasks among the nodes, it is required to enhance the features of customary cameras to include processing and communication capabilities. The processing capability enables the camera module to perform local processing down to pixel level, while communicating features permit information exchange among the camera modules. Assuming that all cameras have signal processing capability and communication media that permits a communication with other cameras and a central unit, omnidirectional vision reconstruction algorithms can be realized in a distributed manner among the camera nodes.

In the distributed and parallel implementation of the Panoptic Camera, each camera constructs a portion of the omnidirectional vision with the help of neighboring cameras. For a distributed implementation of the omnidirectional algorithm, each i^{th} camera must possess the knowledge of its covering directions and the information of the other contributing cameras for all of these directions. This information can be extracted by the internal and external calibration processes of the Panoptic system. After extracting the camera parameters, such as camera direction vectors and coordinates on spherical surface, angle of views (AOV) of each camera, etc., each camera can construct its assigned portion of omnidirectional view inde-

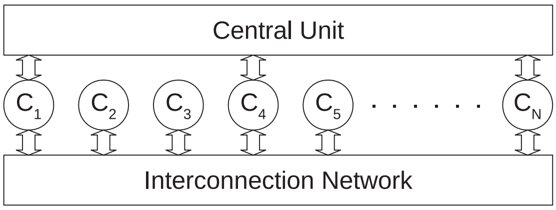


Figure 4.3 – High level model of an interconnected network of cameras. All cameras C_i are connected via interconnection network and some cameras have direct access to central unit.

Algorithm 1 Distributed Reconstruction Algorithm

```
calculate calibration data
calculate weights
for all best observing directions do
    $P_m := read\_pixel\_from\_memory$
    $p_{contr,2..n} := request\_pixels$
    $C := W_m \cdot P_m + \sum_2^n P_{s,n}$
    send $C$ to central unit
end for
for all other observing directions do
    wait for request from principal camera
    $P_s := read\_pixel\_from\_memory$
    $P_{s,out} := W_s \cdot P_s$
    send $P_{s,out}$ to principal camera
end for
```

pendently. The distributed implementation of the algorithm is summarized in Algorithm 1. The required information can be calculated once by the central unit and updated to the local memory of the camera modules. Alternatively, each camera module can calculate its own required information using its own processing features. In the initialization process, the set of best observing directions for each camera is extracted. Furthermore, other contributing cameras for each coverage direction and their weights used in the second interpolation step are extracted. After the initialization process, each camera has the knowledge of which $\vec{\omega}$ to construct, which other cameras are contributing to the same $\vec{\omega}$ and, depending on the interpolation type, what are the camera weights contributing to the final level of interpolation. Assuming cameras have processing capabilities, the missing variables to construct the light field are the light intensity values obtained by the other cameras. This creates the necessity of a communication scheme among the camera modules.

4.1.5 Interconnected Network of Cameras

A general purpose message-passing interconnection network is a programmable system capable of exchanging data between terminals. The system illustrated in Fig. 4.3 shows N

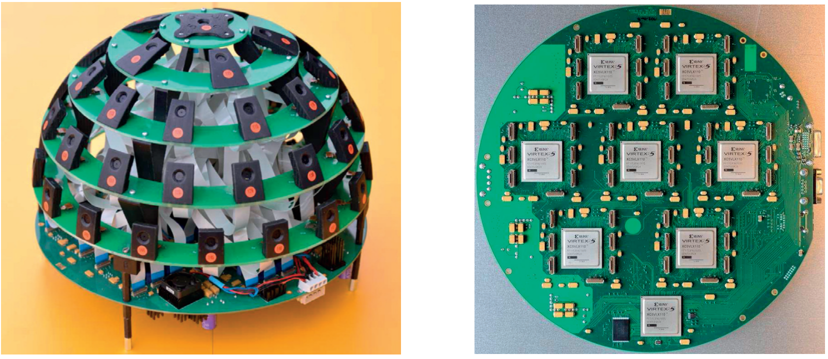

(a) (b)

Figure 4.4 – (a) The network-based Panoptic prototype with 5 floors and 49 cameras. The sphere diameter of the prototype is $2r_{\odot} = 30\text{cm}$. (b) Top view of the Panoptic Media FPGA-based development platform.

terminals, C_1 through C_N connected to a network. As an example, when terminal C_2 wishes to exchange data with terminal C_5, C_2 sends a message containing the data to the network and the network delivers the message to C_5. The terminals C_i resemble the camera nodes with features in addition to basic imaging.

Having a distributed camera system does not imply the omission of a central unit. For example, a central unit is required for the cameras to send their processed information for the purpose of display. It is preferred that all cameras also have a direct access to a central unit. However, this feature is not feasible or optimal in most cases. A central unit may not have enough ports to interface with all the cameras of the system. In case where all the cameras are connected to the central unit with distinct interfaces, and the respective bandwidth of these connections are not fully utilized, such an arrangement may cause inefficient usage of resources. Hence it is more efficient to provide some of the cameras with direct accessing capability to the central unit and share these connections with the cameras that do not have a direct interface to the central unit. The availability of an interconnection network permits the utilization of this strategy. The latter concept is depicted for the Panoptic system with N number of cameras in Fig. 4.3.

In multi-camera applications, information exchange mostly takes part among the neighboring cameras. Thus, during the creation of an interconnection network, neighborhood relations of camera modules should be preserved as much as possible. In order to obtain the neighborhood relation graph of the Panoptic system, the surface of the Panoptic device hemisphere is partitioned into a set of cells centered on the camera locations. Each cell is defined as the set of all points on the hemisphere which are closer to the camera location contained in the cell than to any other camera positions. The boundaries of the cells are determined by the points equidistant to two nearest sites, and the cell corners (or nodes) to at least three nearest sites.

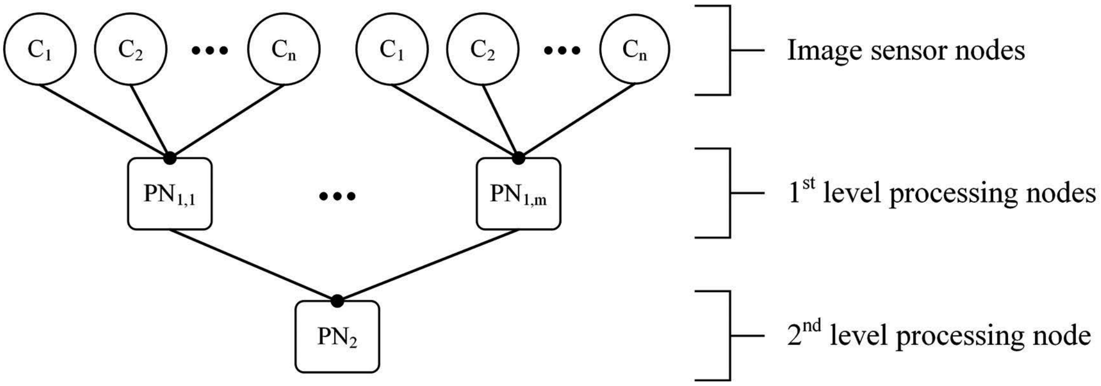
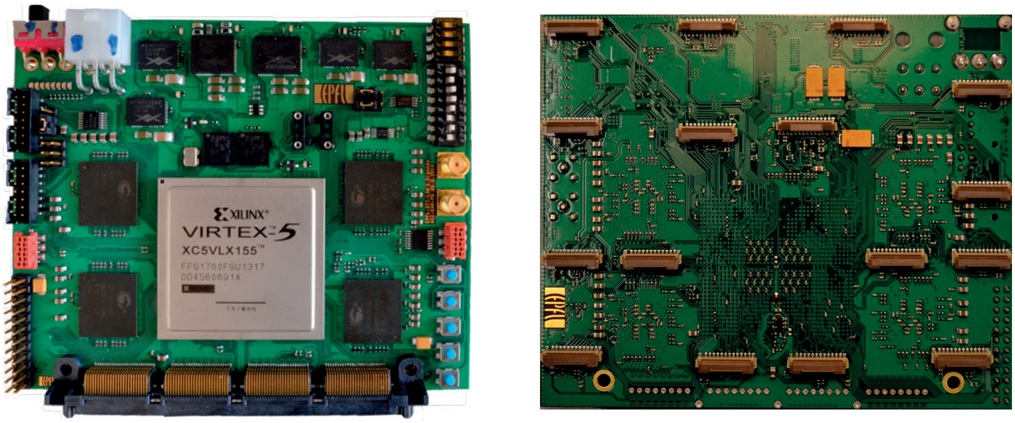


Figure 4.5 – Network with tree topology of depth 2. PN1 operates on images collected by C and PN2 stitch partially reconstructed images by PN1. Maximum supported number of cameras is determined by fanout of PN1 and PN2 (m, n, respectively)

4.1.6 Novel hybrid architecture

As indicated before, memory storage and bandwidth requirements constitute an important challenge for real-time omnidirectional reconstruction. Furthermore, implementation of computer vision algorithms such as feature detection and description, increase the memory requirements much more. The system presented in [17] consists of 49 processing units (PU), where each unit has a memory, a camera unit and image processing part. This feature increase the number of cameras which can be supported by the system, since each camera can act like an omnidirectional image reconstruction system on its own. Furthermore, a communication media is necessary for the data exchange among the PUs. Since each unit has its own image processing part, the same hardware architecture should be repeated. This causes significant increase in terms of resource allocation. Furthermore, the memory bandwidth is not efficiently utilized, since each camera reads redundant data, although they are not responsible of reconstruction. The network-based architecture allows the system to be scalable, but also adds further latency and utilizes hardware resources. Finally, the algorithm and units are designed to be scalable, therefore it is easy to add or subtract cameras from the system such as adding a new nodes to a network, but the proposed hardware architecture is limited with 49 cameras since no additional board or modules can be added. Due to the limitations of the previous systems, we designed a new hybrid panoptic architecture and propose a hardware solution.

In this architecture, similar to the distributed approach, computational load of the omnidirectional reconstruction is distributed between the processing nodes, which have a network with tree topology (see Figure 4.5). In this network, image sensor nodes (cameras) take place as leaves. Processing nodes stitch the images with the algorithm which is explained in Section 2, and consist of the processing units, memories and connectors. The first level processing nodes (PN1) stitch the image data collected from the image sensor nodes (cameras). In the other levels, stitching process is operated on partially stitched images. In fact, a full omnidi-



(a) (b)

Figure 4.6 – (a) Front view of the horizontal board carrying FPGA, memories, power units and misc. devices (b) Back view of the horizontal board carrying 16 camera connectors

rectional image can be reconstructed with a tree of depth 1, with PN1 standalone. However, the sensor nodes which can be supported by PN1, fanout of PN1, is limited with the number of pins and hardware resources of PN1. Therefore, depth level is supposed to be increased, if more cameras needed in the system. Owing to modularity of the architecture provided by the network with tree topology, a wide range of Panoptic system of different number of cameras, can be built by changing the fanout and depth of the tree.

The main processing components of this architecture, PN1 and PN2, can be realized as custom-designed FPGA-based high-performance boards, taking into account real-time constraints. Image sensor nodes, which are the leaves of the tree topology, are VGA cameras with parallel output. Since panorama construction and feature extraction algorithms need very irregular and fast memory access, the main target of the hardware implementation is maximizing the memory capabilities in terms of both memory storage area and bandwidth. Therefore, memories and cameras are connected to FPGA separately in order to increase the bandwidth, thus FPGA pin count limits the number of cameras that can be connected. Because of this limitation, fan-out of PN1 is 16, which means that each PN1 board has 16 camera connectors. While the connection between C and PN1 nodes is provided by flexible parallel cables, 90 degree data connector is placed for PN1 and PN2 communication. Thus, PN1 boards can be mounted vertically onto PN2 boards. Xilinx Virtex-5 FPGA is used as the processing unit of PN1, due to its parallel processing capability, high amount of internal resources and pin count. Memory units consisting of 4 Quad Data Rate (QDR-II) SRAMs capacity of $8M \times 18bit$ which are separately connected to FPGA, provides 32 Megapixel (MP) storage area and 1600 MP/s bandwidth, assuming each pixel data is stored in 18-bit memory cell. Consequently, PN1 provides a memory space to store 6.75 VGA frame per each camera. The nature of the QDR-II SRAM which has separate input and output ports and data transfer both falling and rising edges of the clock, improve the bandwidth 4 times, however a new approach on the algorithm

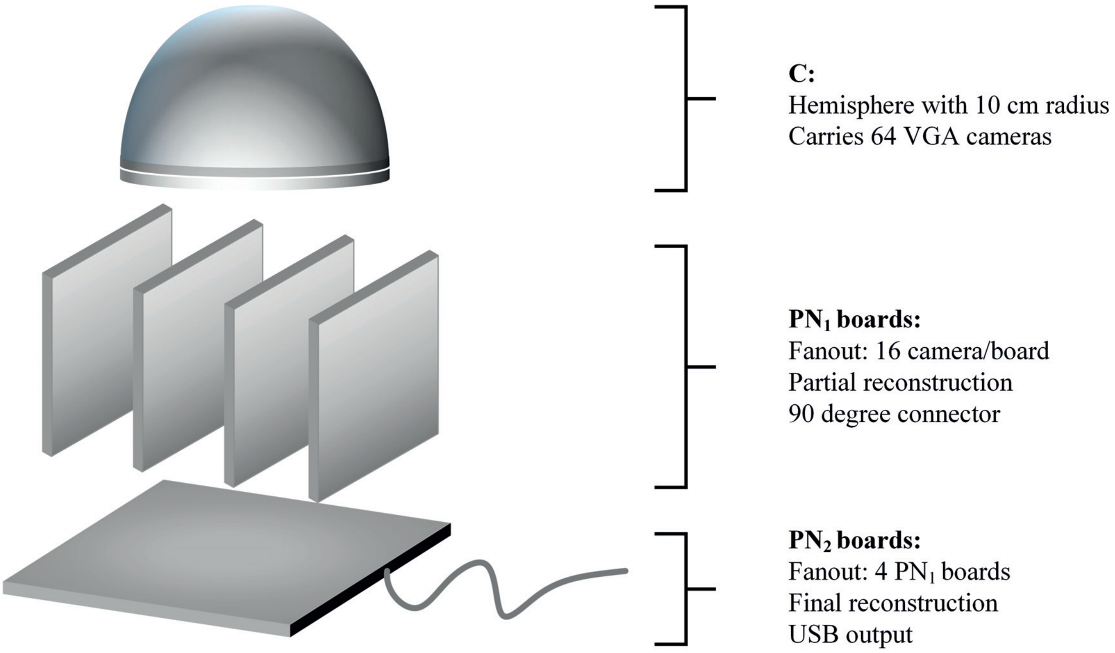


Figure 4.7 – Hardware solution for a system which has single PN2. Cameras is placed on hemisphere, and collected images are transmitted to vertical boards which represents PN1. Omnidirectional image is completed on horizontal board, PN2, processing of stitched image by PN1.

is needed in order to benefit this performance increase. A comparison with previous panoptic systems can be found in Table 4.1. Manufactured and assembled PCB is shown in the Figure 4.6.

	Storage per camera	**Total storage**	**Total bandwidth**
Ping-pong [16]	2 CIF frame	4 MP	670 MP/s
P-49 [17]	3 VGA frame	49 MP	4900 MP/s
Proposed PN1 board	6.75 VGA frame	32 MP	1600 MP/s
Proposed system	6.75 VGA frame	128 MP	6400 MP/s

Table 4.1 – Storage and bandwidth comparison of the Panoptic systems

The tree node PN2, can be realized as a horizontal board, which PN1 boards are mounted vertically. Fanout of PN2 is restricted with pin count of the FPGA, and equal to 4. By using proposed hardware (C, PN1, PN2), it is possible to generate a number of different solutions by changing connections. Since fanout of PN1 is 16, and fanout of PN2 is 4, a system which has one PN2 supports up to 64 camera and an example structure can be seen in Figure 4.7. If more camera is needed, either depth of the tree is increased or PN2 boards are connected with a different network topology.

4.2 Polarimetry

Polarimetric imaging is an emerging remote sensing technology which complements panchromatic, multi and hyper-spectral imaging. While spectral signatures carry information about material properties, the polarization state of an optical field depends on surface features, such as shape and roughness. The information acquired by spectral and polarimetric imaging often reveals independently distinctive features of relevant objects. With such capabilities, polarimetric imaging demonstrates enhanced competencies for advanced object detection tasks [86], especially with contrast-enhancing techniques.

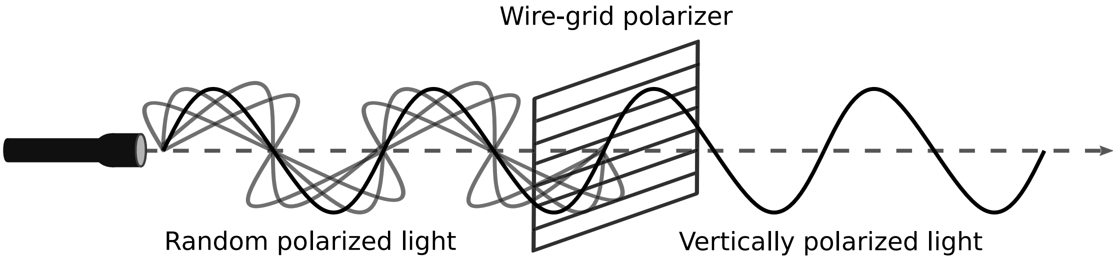


Figure 4.8 – Horizontally placed wire-grid polarizer absorbs the horizontal components.

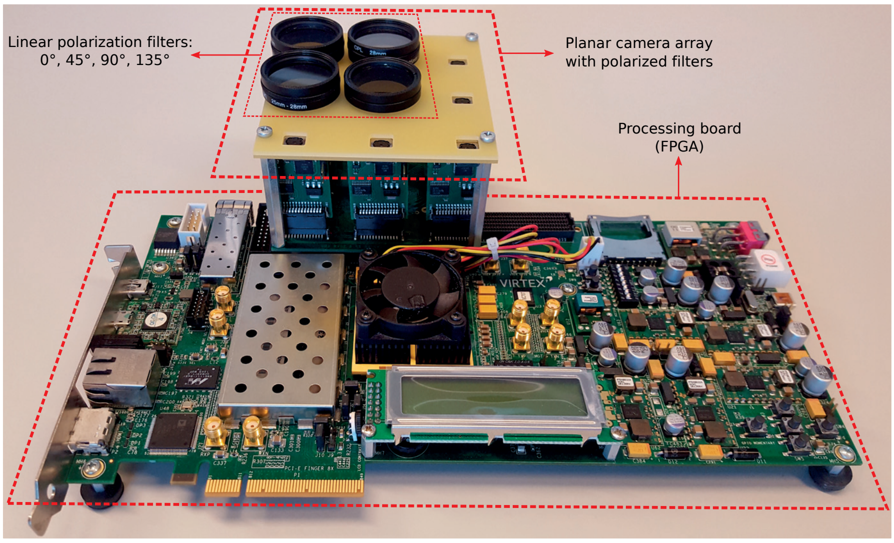
In more details, polarimetry is the measurement and interpretation of the polarization state of transverse waves, such as radio or light waves. Typically it is done on electromagnetic waves that traveled through or reflected by some material in order to characterize that object [87]. Polarimetry is proved to be beneficial for numerous visual enhancement and scene identification methods. To remove the effects of haze from images, recover visibility of underwater scenes, improve the contrast and separate the specular components in an optical field, polarimetric imaging techniques demonstrated an outstanding performance [88]–[90]. Moreover, polarimetry is often used during post processing of various target detection applications to improve the characterization of a target by estimating the texture of a material, resolving the orientation of structures [91].

It is often preferred to represent the polarization information in terms of the Stoke vectors, which is defined as a time-averaged intensity measurement [86]. In order to compute the elements of the Stoke vectors, filtered images by special polarizing filters are needed. A filtering example is illustrated in Figure 4.8. The wire-grid polarizer absorbs the horizontal components and allows only vertical polarization to pass. Polarimetric cameras used in this study have linear polarizing filters placed with different angles. This enable us to compute the first three elements of the Stoke vectors: S_0, S_1 and S_2 from the output of the cameras. We preferred not to use Stoke parameter S_3, calculated by left and right circular polarizations which are usually quite weak for natural backgrounds. The usage of filters is drastically different between these devices.

The first one is called Equus 327k SM (Figure 4.9a), which is an infrared camera in SW/MWIR spectra and developed by IRCAM. It is based on a mechanism that circularly rotates filters

(a) Equus 327k SM

(b) PolarCam-V



(c) Custom design multi-camera polarization

Figure 4.9 – Polarization camera examples

corresponding to different polarization components. As an obvious downside of such systems, since there is a delay between the consecutive frames, stoke parameters of the pixels belong to moving objects cannot be computed correctly.

To overcome this issue, we used another polarimetric camera by 4D Technology named PolarCam V in VIS spectra, shown in Figure 4.9b. It is a snapshot micro-polarizer camera which is capable of simultaneously capturing multiple polarization angles of an image. More specifically, it uses a technology called wire grid polarizer in which nano-scale patterning is utilized to form a metal grating on a glass substrate [92]. It contains a pattern of four discrete polarizers with 0, 45, 90, 135 degrees. This structure is known as a super pixel and can be seen in Figure 4.10. A more detailed look at technical specifications regarding both conventional and polarimetric systems used in this study is provided in Table 4.2.

Although PolarCam has solved simultaneous capturing problem, the special manufacturing process makes it cost inefficient. A multiple camera system operates synchronously can also provide polarizing information simultaneously. In Figure 4.9c, our hardware proposal is shown. Four of the cameras are covered by linear polarization filters, placed in four different directions: 0°, 45°, 90° and 135°. These cameras are connected to a powerful FPGA development board VC707 with Virtex-7 FPGA via its FMC connector. Thus, camera data filtered by linear polarization filters arrives at FPGA, and DoLP data can be computed in real-time by FPGA. By this way, high resolution can be reached at low-cost.

Table 4.2 – Details regarding to the camera systems used in this study

Camera specifications					
Name	**Sensor**				
	Spectra	Sensor	$\lambda[nm]$	Pixel	Pitch $[\mu m]$
Polarcam V	VIS Polar.	Si	400-700	640x460	7.4
Equus 327k	SW/MWIR	MCT	1500-5000	640x512	15
Canon D5 Mk	VIS	Si	400-700	5760x6840	6.25

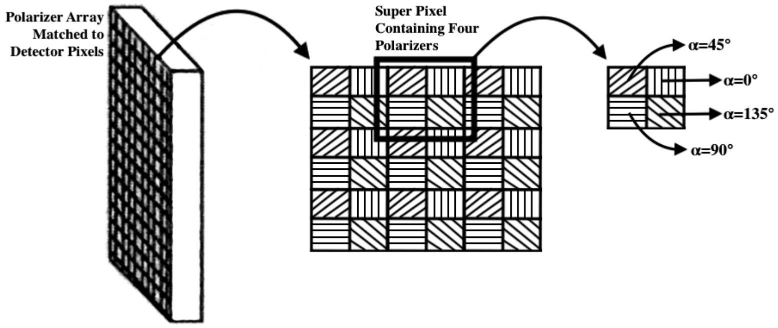

Polarizer Array Matched to Detector Pixels
Super Pixel Containing Four Polarizers
α=45°
α=0°
α=135°
α=90°

Figure 4.10 – Super pixel structure of an image obtained by the Polarcam V [92]

As for object detection systems, they were able to achieve outstanding performance in recent years. Deep learning, in particular, has emerged as the most powerful model for advanced object detection tasks. As the name implies, these models have deep network architectures capable of discovering highly complex patterns. This characteristic, alongside with robust training methods, allow deep learning models to construct powerful object representations by building high-precision part-based models for a variety of object classes [93]–[95].

There are various object detection models that are highly in use and considered as state-of-the-art models. Briefly explaining, Faster Region-based Convolutional Network method (Faster R-CNN) builds on region proposal algorithms to hypothesize object locations. It introduces a Region Proposal Network (RPN) that shares full-image convolutional features with a detection network that enables cost-free region proposals. RPNs are trained end-to-end to generate high-quality region proposals, which are used by Fast R-CNN for detection [96]. In contrast to Faster R-CNN, Region-based Fully Convolutional Networks (R-FCN) model is fully convolutional with almost all computation shared on the entire image instead of applying a per-region sub-network hundreds of times [97].

Similar to R-FCN, Single Shot Multi-box Detector (SSD) also utilizes a single deep neural network for an entire image. SSD is simpler in a sense that it completely eliminates proposal generation and subsequent pixel or feature resampling stages and encapsulates all computation in a single network which makes SSD easy to train and straightforward to integrate into systems [98]. Its uncomplicated nature and competitive accuracy in advanced object detection tasks motivated us to use SSD as the main object detection model in this study. Further information and experimentation of object detection models are provided in following sections.

Despite numerous developments in object detection, computer vision industry is still facing challenges in terms of accuracy and speed as the amount of data increases. The complicated nature of conventional imaging raises difficulties for deep learning models to operate effectively. Various technologies have been developed to improve signal-to-noise ratio and multispectral imaging to obtain spectral characteristics. However, higher contrast is required based on the physical properties of materials. The polarization state plays a key role here as it holds descriptive information about surface properties.

In this study, the primary objective is to utilize the state-of-the-art deep learning models designed for object detection tasks with images obtained by polarimetric systems. With the additional information that polarimetric imaging reveals, it was expected for object detection models to improve performance. We started with the construction of suitable polarimetric imaging dataset using imaging equipments mentioned above. Then, we generated various object detection models with images obtained by both polarimetric and conventional techniques. Finally, we presented the best performing models in various scenarios and proposed a method to achieve the highest accuracy.

In this section, firstly we present the procedures and methods that we applied, and justify the choices. In addition we propose a multiple camera hardware solution for polarimetry applications. Then finally we conclude the section by describing the results and related discussions.

4.2.1 Method

Methods followed in this study can be divided into five steps. We started with the construction of the dataset using both polarimetric and conventional imaging techniques. Then, certain transformations are applied to the raw polarimetric images in order to properly utilize polarization components. This section is indicated as Transforming Images in Figure 4.11 where S_0, S_1 and $DoLP$ corresponds to various transformations explained in Section 4.2.1. In the third step, we labeled relevant objects in acquired images in order to create training samples for our object detection models. We generated various models by using state-of-the-art network architectures and combinations of different polarization components. Finally, we measured and compared performances of trained models which is the section located at the bottom of Figure 4.11. As before, S_0, S_1, $DoLP$ are placeholders for images with corresponding transformations. The overall picture of the methodology followed in this study is illustrated in Figure 4.11. Further details regarding individual steps are explained in the following subsections.

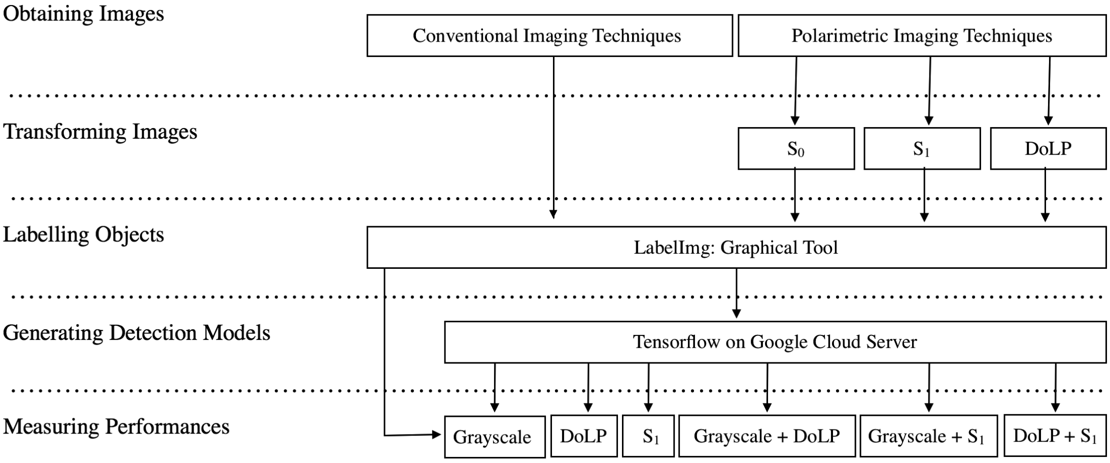


Figure 4.11 – Overall look at the methodology followed in the study

Dataset Construction

At the time of this project, there was no publicly available polarimetric imaging dataset. Hence, the initial task was to acquire adequate data by using polarimetric imaging systems including Equus 327k and PolarCam V which can be utilized to infer the polarization state of an optical field. We collected around 200 images of military vehicles from 30 different locations. These numbers double with the data priory obtained by armasuisse itself. The content of these images includes various types of military vehicles taken from different angles and located in various backgrounds. Their distance from the camera also differs considerably. Sample raw images can be seen in Figure 4.12.

Image transformation using polarization components

As mentioned, an image obtained from polarimetric cameras put in use contains a pattern of polarizers with four discrete orientations corresponding to 0, 45, 90 and 135 degrees. In order to utilize features of such polarimetric images in object detection systems, it is required to make certain transformations on raw images obtained directly from the cameras. According to previous researches [86], Degree of Linear Polarization (DoLP) can be benefited to emphasize man-made objects and depreciate natural ones. Since this property is highly suitable for the purpose of this study, we decided to use DoLP to transform the raw images. It is calculated

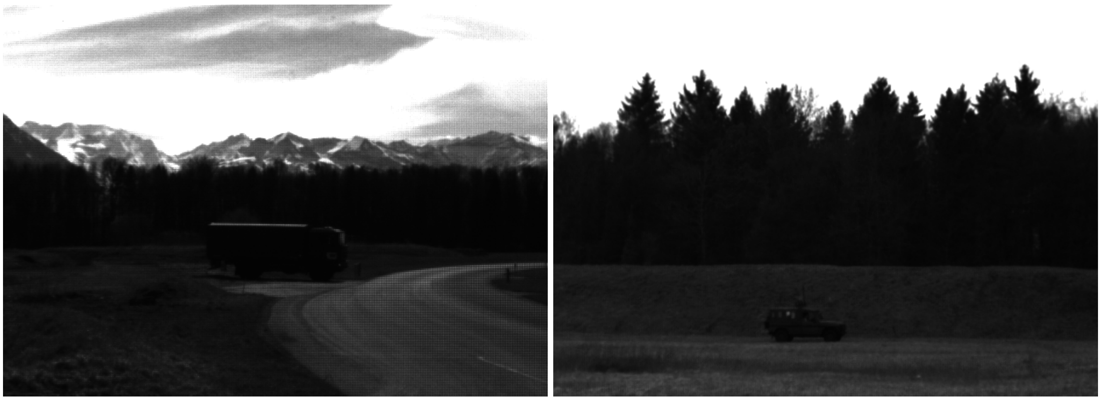


Figure 4.12 – Raw polarimetric images of various military vehicles within different backgrounds

using polarization components as in the following equations:

$$\begin{bmatrix} S_0 \\ S_1 \\ S_2 \\ S_3 \end{bmatrix} = \begin{bmatrix} I_0 + I_{90} \\ I_0 - I_{90} \\ I_{45} - I_{135} \\ I_{LHC} - I_{RHC} \end{bmatrix} \tag{4.3}$$

$$DoLP = \frac{\sqrt{S_1^2 + S_2^2}}{S_0} \tag{4.4}$$

Here, $I_0, I_{90}, I_{45}, I_{135}$ are the intensities of the linear polarization components (polarizers) corresponding to 0, 90, 45, 135 degrees. Since raw images obtained from polarimetric systems used in this study were different, we followed different procedures while transforming them. As explained, the wire grid polarizer in Polarcam V contains a pattern of four discrete polarizers with 0, 45, 90, 135 degrees known as super-pixel. In order to obtain images purely formed by each polarization components, we transformed the raw images using super-pixel structure. For instance, in order to obtain an image only consisting of polarization parameter S_0, we added the I_0 and I_{90} polarizers located in each super-pixel. This value is calculated for each super-pixel and entire image for polarization parameter S_0 is constructed by putting these value to corresponding pixel locations. As for IRCAM camera, this procedure was relatively straightforward since we obtained four different images for each discrete polarizer, instead of a single image containing all four. S_0 in this case is calculated by using pixels in two separate images corresponding to I_0 and I_{90} polarizers. After obtaining an image for each polarization components that are S_0, S_1 and S_2, we were able to directly calculate DoLP images using Equation 4.4. In addition to DoLP, we also investigated the performance of individual polarization components within object detection systems by considering them separately.

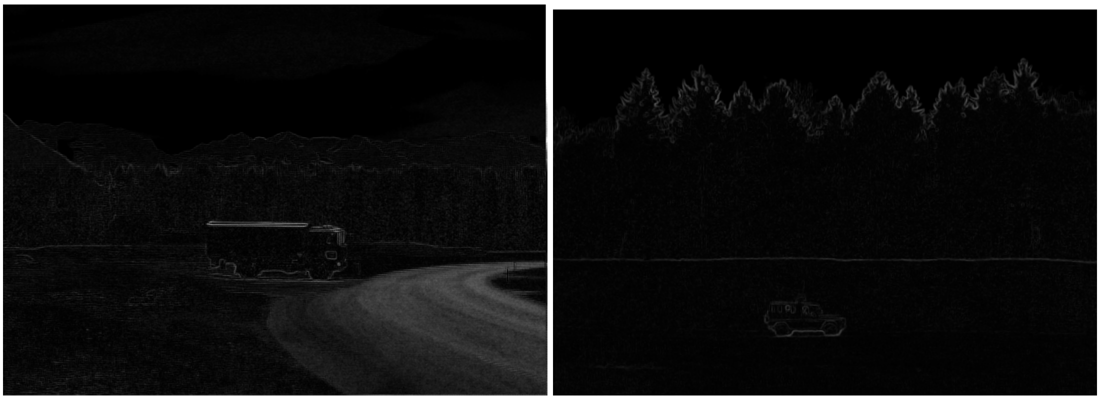


Figure 4.13 – Polarimetric images after transformation with DoLP

It is evident that transformation with DoLP emphasized only man-made objects that are military vehicles while understating irrelevant parts of images including mountains, trees, and grass as shown in Figure 4.13. This property is expected to be quite beneficial for object detection systems since relevant objects become strongly distinguishable after the transformation.

Object Labeling

The exact location of objects within images are required to be specified for the training phase of object detection models. We manage to do this operation manually by using an application named as LabelImg [99]. It is a graphical image annotation tool written in Python and uses Qt for the graphical interface. In our case, we are interested in detecting military vehicles and therefore, only labeled such vehicle in acquired images.

Depending on the amount of collected data, manually selecting and labeling each object might not be a feasible. It was an applicable method in our case, however, other ways of automating this process, such as taking advantage of pre-trained masking models should be considered for bigger scale projects.

Training and Testing of Object Detection Models

Initial experiments with pre-trained models which are built on various deep learning architectures including Faster-RCNN and SSD, reemphasized the requirement of training custom models to be able to comprehensively utilize the additional benefits that polarimetric imaging provides. As seen in Figure 6, pre-trained object detection model (which is based on Faster-RCNN) was able to detect the vehicle in the gray-scale image (on the left) with much higher confidence even though, the vehicle is more exposed and distinguishable in the polarimetric image. This is a consequence of the fact that pre-trained models encountered similar gray-scale images during their training phase.

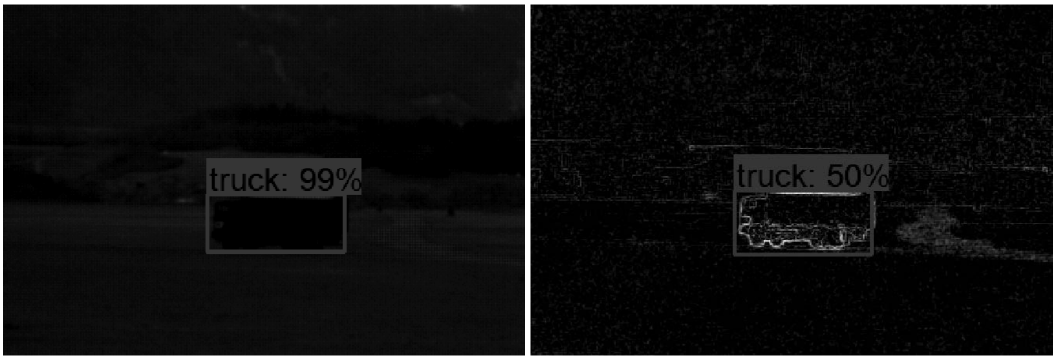


Figure 4.14 – Detected objects in polarimetric and gray-scale images using pre-trained models

As mentioned earlier, we utilized the SSD deep learning architecture for object detection tasks conducted in this project. Comparing to other state-of-the-art models, SSD eliminates various redundant steps and allows proper training with fewer sample images without compromising from detection performance. Considering the amount of polarimetric image data within our reach, SSD was the best-suited model for our purposes.

After completing the object labeling step, we split the available data into training and testing parts as 60% and 40% respectively. In order to increase the amount of training sample, we took advantage of various data augmentation techniques including randomly flipping the orientation or cropping of certain parts of an image. Even after augmenting, the data we possessed was not sufficient to train a robust object detection model from scratch. Hence, we decided to start the training processes from a checkpoint which corresponds to a model trained up to a certain point with weights still highly susceptible for any updates. We train the model using our data where the checkpoint left off. The obvious shortcoming of this approach is that the checkpoint models are trained with RGB and gray-scale images and not with polarimetric ones. This will cause noticeable positive performance bias toward gray-scale images during the testing phase.

We trained two primary models by using gray-scale images and polarimetric images transformed with DoLP. To investigate the effect of an individual polarization component, we also trained models just by using one of the components, namely S_1. Additionally, several others models were trained by combining gray-scale with DoLP, S_1 with DoLP and gray-scale with S_1. In total, we manage to train 6 different models.

As the machine learning framework of choice for the project, we used TensorFlow [100] which has numerous built-in features and models including Faster R-CNN and SSD. Storage and memory requirements to train a deep neural network is often exceeds the capacity of a single machine and therefore, a cluster consisting of multiple machines is necessary for training phase. We decided to use Google Cloud Services (GCS) for that purpose. It supports TensorFlow and offers convenient ways of utilizing multiple machines for deep learning tasks. The cluster used in this project consisted of 8 worker machines with GPUs of 3.75 GB memory. The average duration of the training phase was 1 hour 45 minutes and memory utilization of workers during this process was approximately 40%. Within this settings, it took around 5000 steps to train each model. Loss function becomes stable and prediction accuracy no longer increased afterward as demonstrated in Figure 4.15.

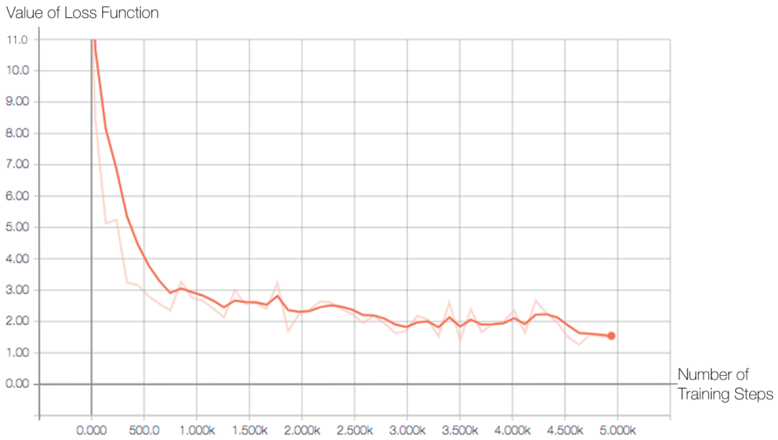


Figure 4.15 – Value of the loss function with respect to number of steps during the training phase

Memory space occupied by a single trained model was around 160MB and node weights were varied between values -18 and 46. As for testing, it is done on a local machine with 2.5 GHz CPU and an 8 GB of RAM. It took 4.75 seconds on average for trained models to process a single image and detect objects within.

4.2.2 Result

For the testing phase, different images with various military vehicle and backgrounds were used. The performance of the model trained with polarimetric images transformed by DoLP was satisfactory. It manages to achieve similar performance with models trained using grayscale images that are obtained by conventional imaging techniques. Our experiments with the polarization parameter S1, on the other hand, did not reach desired performance. Accuracies that trained models managed to achieve are presented in Table 4.3.

The accuracy of the model trained with grayscale images was the highest, followed by models trained with polarimetric images transformed using DoLP. It is likely that the better performance of grayscale is a result of using checkpoints. As mentioned earlier, these checkpoints are constructed using grayscale and RGB images and biased towards model trained with grayscale images. Even this being the case, models trained with DoLP were still able to achieve

Table 4.3 – Accuracy of the trained models

Model accuracies			
Trained with	**Tested on**		
	Grayscale	DoLP	S_1
Grayscale	81%	-	-
DoLP	-	75%	-
S_1	-	-	30%
Grayscale + DoLP	75%	76%	-
Grayscale + S_1	65%	-	37%
DoLP + S_1	-	13%	15%

significantly higher accuracy which indicates that training an object detection model from scratch with couple thousand polarimetric images is likely to outperform current object detection model trained by conventional imaging techniques. Moreover, we observe that DoLP trained models were able to demonstrate even better performance in certain scenarios. If an image is sharp with objects that are easily distinguishable from the background then, gray-scale trained model detected them with a strong confidence. On the other hand, for blur images with objects that disappear in the background, the DoLP trained model achieved significantly higher confidence levels which can be observed in Figure 4.16. Also note that in order to calculate confidence levels, SSD generates different bounding boxes and adjust these boxes as part of prediction. It finds a score for automatically generated bounding boxes with various shapes and sizes. Then, it tries to keep bounding boxes with high scores meaning that probability of an object being in that box is high. The percentages in Figure 4.16 represent these scores.

For the image at the top, detection with gray-scale trained model reached a better confidence percentage since the vehicle is easy to distinguish from the background. The situation was vice verse for the bottom image and therefore, DoLP trained model demonstrated better performance. With this observation, we decided to run these models in parallel with each other. For the scenarios where one of the models lack, the other one was able to compensate for that. By doing so, we manage to achieve an accuracy score of %90 which was the highest among all models tested in this study. These results suggest that the performance of state-of-the-art object detection models can be exceeded and considerably improved by using images obtained by polarimetric imaging techniques.

4.2.3 Conclusion

Object detection systems using advanced deep learning model were able to achieve exceptional performances in recent years. They were able to unfold the intricate structure of objects

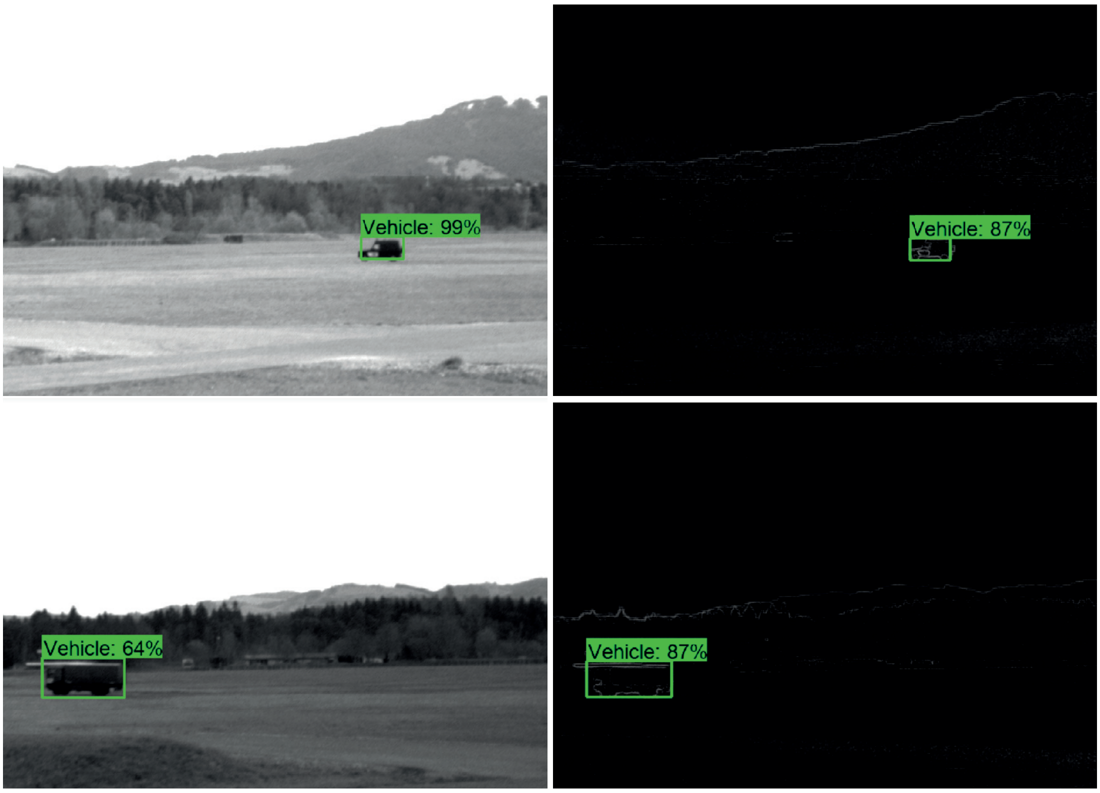


Figure 4.16 – Objects within different backgrounds detected by the model trained with grayscale and DoLP images

by analyzing complex patterns. However, conventional imaging with its complicated nature proved to be troublesome for these detection models. Polarimetric imaging techniques, on the other hand, demonstrated enhanced capabilities for object detection tasks with its capability to discriminate man-made objects from natural background surfaces. In this study, the performance of polarimetric imaging on advanced object detection tasks was investigated further. With additional information provided by these images, we were able to achieve significant performance improvements on state-of-the-art object detection models. We observed that utilizing both conventional and polarimetric imaging techniques alongside with each other demonstrated the best performance.

There are couple of directions to proceed from this study. In parallel with the scale of this project, the amount of polarimetric data we acquired were limited. We were not able to train an object detection model from scratch and started from a checkpoint. Increased performance and more reliable results might be achieved by collecting more polarimetric images and following steps suggested in this study. Additionally, optimizing deep learning models specifically for polarimetric imaging with various methods, such as parameter fine-tuning is likely to increase the performance and might be another improvement and direction to proceed from this study.

4.3 Vision-based surveillance for drone detection

Unmanned aerial vehicles (UAV) or *drones* are autonomous or remote-controlled aircrafts. Although its history goes back to unmanned balloon carriers used in a war in 1849 [101], and mostly used in military through the history, its civil application areas are rapidly expanding in the recent years. The latest developments in production processes of the drone technology make them more affordable, and number of actually operating drones in the air has been exponentially increased. Approximately 300,000 drones are being sold every month, and it is estimated that there will be 5 million drones will be flying in the sky in the next few years. Although this technology is being efficiently used in several areas, such as agriculture, cartography, transportation, search and rescue, bringing improvement in the corresponding tasks, it also causes security and privacy concerns. As an important fact, the drones have much flexible uses and can be equipped with cameras, explosives, and so on for spying, terrorist attacks, etc. purposes. A mechanism that detects in real-time the drones violating the unpermitted civil and military areas could help preventing malicious acts by drones.

Their dimensions of a drone vary from insect-size to 15 m wingspan. Small drones are of carrying high amount of explosive materials whereas big drones are able to carry, but the former is also easy to detect by the current surveillance systems. Middle sized drones capable of carrying dangerous material and could serve malignant actions, however, are difficult to detect. In addition to this, since drones move in 3-dimension mostly without obstacles, they can rapidly get closer to the target. This also brings more strict constraints for real-time detection of middle-sized drones.

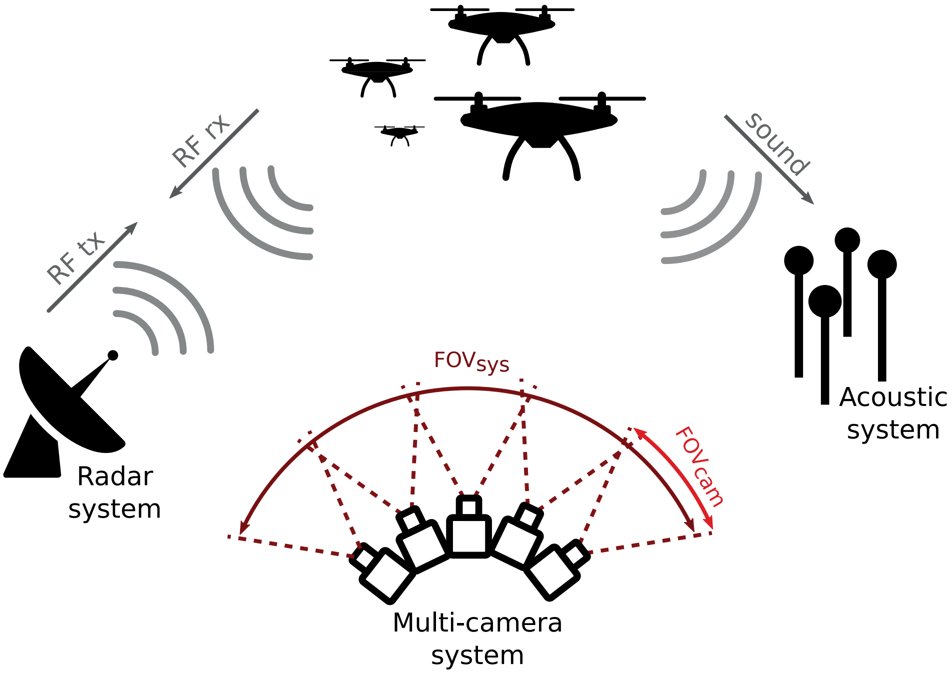


Figure 4.17 – Drone detection methods

Since drones have been a security threat which has not still a feasible and affordable solution, this problem have drawn attention from both academia and industry in the recent years. Attempts to this problem mostly benefit from three types of sensors: acoustic, RADAR and optics [102] as illustrated in Figure 4.17. Acoustic systems composed of microphone arrays have been used to analyze the sound produced by the drones and find the direction of them through beamforming. Although it is reported that the range can be reach up to around 700 m [103], background noise level in urban areas and rural areas differs and the range reduces down to 200 m level or below [104].

Principle of RADAR-based systems relies on analysis of transmitted signal and its received version reflected from objects. They can operate in day and night-time, and are not affected by weather and environmental conditions such as rain, fog, dust or snow. However, in order to detect small sized objects from long range, short wavelength radar signals, such as mm-wave, has to be used but the use of such radars are costly.

Vision-based approaches have been also introduced in the literature. In [105], a boosted cascaded classifier with Haar-like features, Histogram of gradients (HoG) and Local Binary Patterns (LBP) was proposed for UAV detection with an integrated distance estimation mechanism. They concluded that boosted cascaded classifier with LBP has better near—real-time detection performance. Another detection technique was presented in [106] that suggests a multi-staged framework incorporating AdaBoost method along with a tracking algorithm. The method achieves accurate detection meeting real-time constraints. In [107] has been proposed a method combining both motion and appearance information to be used within a machine learning set-up based on AdaBoost, Deformable Parts Model and Convolutional Neural Network (CNN). To improve the performance of CNN-based approach, deep domain adaptation was additionally proposed. A latest work in [108] exploited the distinct motion signature of UAVs through video sequences for the UAV detection problem. Good detection performance has been reported for the UAVs in several status of UAV, such as moving at high or low speeds.

In this problem, hybrid solutions to combine strength of different sensors are also available. In [109], acoustics, RADAR, LIDAR and optical sensors are used together. Another hybrid solution was provided by commercial product Drone Detector where detection up to 1 km range is promised with RF detection [110].

Drones are generally challenging to discriminate from birds. A number of works therefore focused on the bird-drone discrimination. In [111], a nearest neighbor classifier on polarimetric scatterings has been proposed for classifications of drones and birds. An analysis of monostatic and bistatic radar measurements of multiple birds as well as a quadcopter micro-drone was given in [112]. Alternatively, a single shot object model was presented in [113]. Even though, all those attempts provide promising approaches to the this task, the problem still remains challenging to distinguish drones from birds that are far from the source, especially from a vision-based perspective. Accordingly, a Drone-vs-Bird Challenge was launched in conjunction with 14th IEEE International Conference on Advanced Video and Signal based surveillance.

Although vision sensors have usually a complementary role, high-resolution multiple camera systems can play a significant role in drone detection problem. Visual data captured by a static and high-resolution camera system enables to detect even small changes occurred in the field of view. In this section, we propose a solution for vision based drone detection. High resolution cameras enables detection from the long distances. By applying the smart camera concept, reducing the bandwidth makes possible to process all the data locally on the chip, and transmit to the PC only the area which is possible to have a salient motion. This also enables to apply machine learning applications on the software side in real-time.

4.3.1 System overview

Detection range and response time are the most important parameters in the drone detection systems. In order to extend the range in the vision based systems, high resolution sensors are necessary. In addition to this, limited angle of view of a single camera is not sufficient for a full observation. Therfore, multiple of the high resolution cameras are needed.

Considering these requirements, in this work, we used the GigaEye-II (see Figure 4.18) which is a scalable multiple camera system consists of 20-megapixel cameras, specifically CMV20000 from CMOSIS. It is designed to produce high resolution (9000 × 2400 pixels) panorama [114] at 30 fps. Computation of the panorama is performed by the FPGA boards (VC707 and VC709 from Xilinx) of the system, by benefiting of the DDR-3 memories and the high speed connectivity of the FPGA boards.

The connectivity of the system is shown in Figure 4.19. 4 high-resolution cameras are connected to a cluster board of VC709 with Virtex-7 FPGA from Xilinx via its FMC connector. A concentrator board can serve 4 of the cluster boards and 16 cameras in total. The connections between concentrator boards and cluster board are provided by optical links.

This system aims at creating panorama and apply a drone detection method on the omnidirectional image. Operations on the panorama prevents jumping of the target from one camera to another camera, it is always remains on the composite image. This makes the tracking of the targets easier. However, during the panorama construction, pixels are collapsed in the panorama. In GigaEye-II, $16 \times 20 = 320$ MP data is captured, but in the panorama at $9 \times 2.4 = 21.6$ MP is represented. Although a significant portion of this data comes from overlapping area and does not carry different information, when the target drone is at the long distance, it occupies a few pixels and each pixel is important. These small targets may disappear in the panorama. Therefore, evaluating each camera separately enables to extend the range.

Maximum distance of the target that can be detectable by the software is calculated by the pin-hole camera model, and similarity of triangles. Assuming the software is able to detect

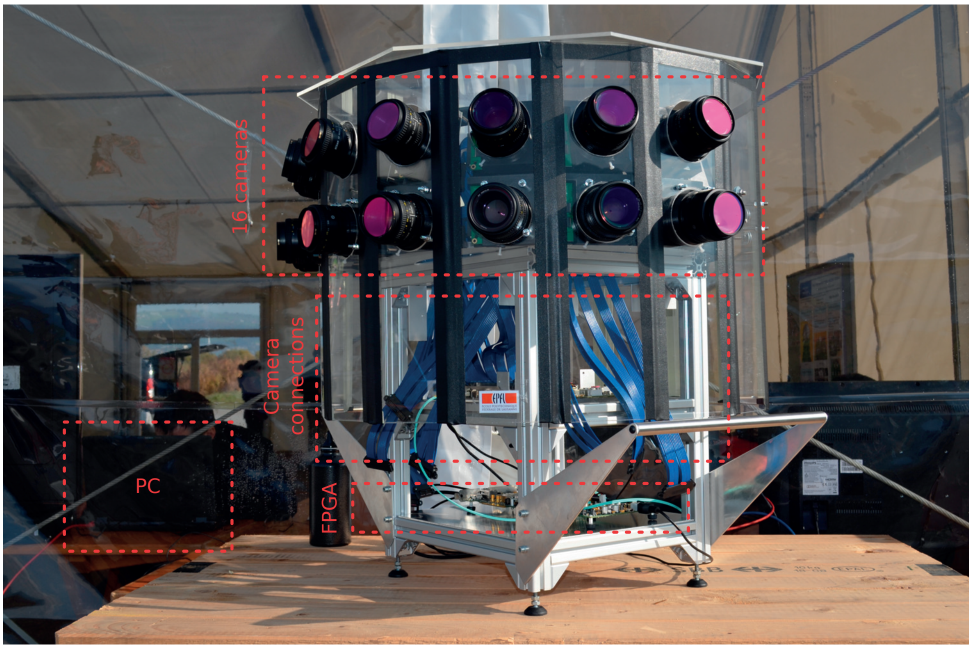


Figure 4.18 – GigaEye-II, captured in drone detection days of armasuisse, 2016

a moving object wider than 1 meter in horizontal and its corresponding shape on the image sensor has at least 10 consecutive pixels, maximum distance is given in the following equation:

$$D = \frac{f \times h_{\text{im}}}{h_{\text{re}}} = \frac{50\text{mm} \times 1\text{m}}{10\text{px} \times 6.4\mu m} = 780m \tag{4.5}$$

Here, the lens focal length ($f = 50mm$) and the image sensor size ($h_{im} = 6.4\mu m$) is fixed in the GigaEye system. However, this range estimation is valid under the specified assumptions, and could show difference depending on the software algorithm.

In order to reach this detection range, each camera should be evaluated separately. On the other hand, this operation brings a challenge in terms of bandwidth, due to very high resolution of the full system. The total bandwidth generated by the camera sensors can be calculated as follows:

$$\text{BW} = 16 \text{ cam} \times 20 \text{ MP} \times 24 \text{ bit/px} \times 30 \text{ fps} \approx 230 \text{ Gbps} \tag{4.6}$$

Since this number exceeds the PCIe and ethernet limits, it is not possible to transmit all the data to a single PC. A better solution which increase the bandwidth capacity of the system is depicted in Figure 4.20. Differently from the system in Figure 4.19, cluster boards are directly connected to the PC with PCIe connection. For the PCIe connectivity of this design, we benefit from the RIFFA framework [115].

In this solution, although bandwidth capacity of the system is increased, frame rate is sacrified

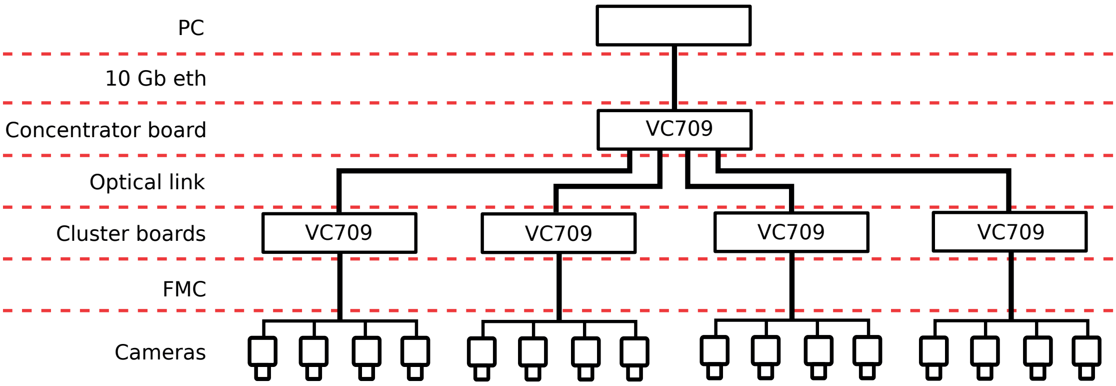


Figure 4.19 – GigaEye-II connectivity with single PC via 10 Gb ethernet

in order to overcome the bandwidth limitations. Since it is not possible to transmit the data from all the cameras at the same time, frames are sent sequentially to the PC. On the other hand, based on the fact that increasing the level of intelligence of the cameras reduce the bandwidth, applying the smart camera approach in the GigaEye system makes it possible to meet real-time requirements overcoming the bandwidth limitations.

Proposed architecture with smart camera is shown in Figure 4.21 Moving object detection blocks use the first DDR-3 memory to store the background models of the cameras. Parallel to the pixel stream, they generates the metadata of the positions of the moving objects. This metadata information is used to select the camera and its sub-window which contains the moving objects, and this pixel information is directed to the second DDR-3 memory. In parallel to this, FIFO in the RIFFA PCI interface is filled by this data. When the software in the PC requests this data it is transmitted via FIFO.

Display operation on the PC side is shown in Figure 4.22. Central screen is dedicated for composite view and control panel. In case of moving objects are found in the field of view, their corresponding sub-windows are displayed in the left and right screens with the full resolution. The targets are marked and their trails are drawn.

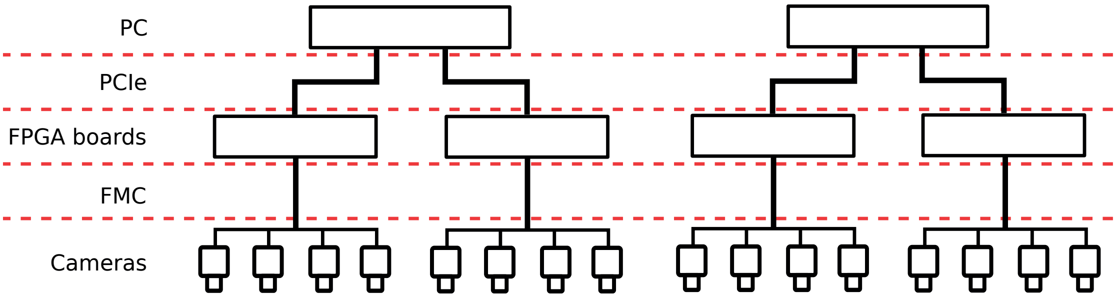


Figure 4.20 – GigaEye-II connectivity with dual PC via PCIe

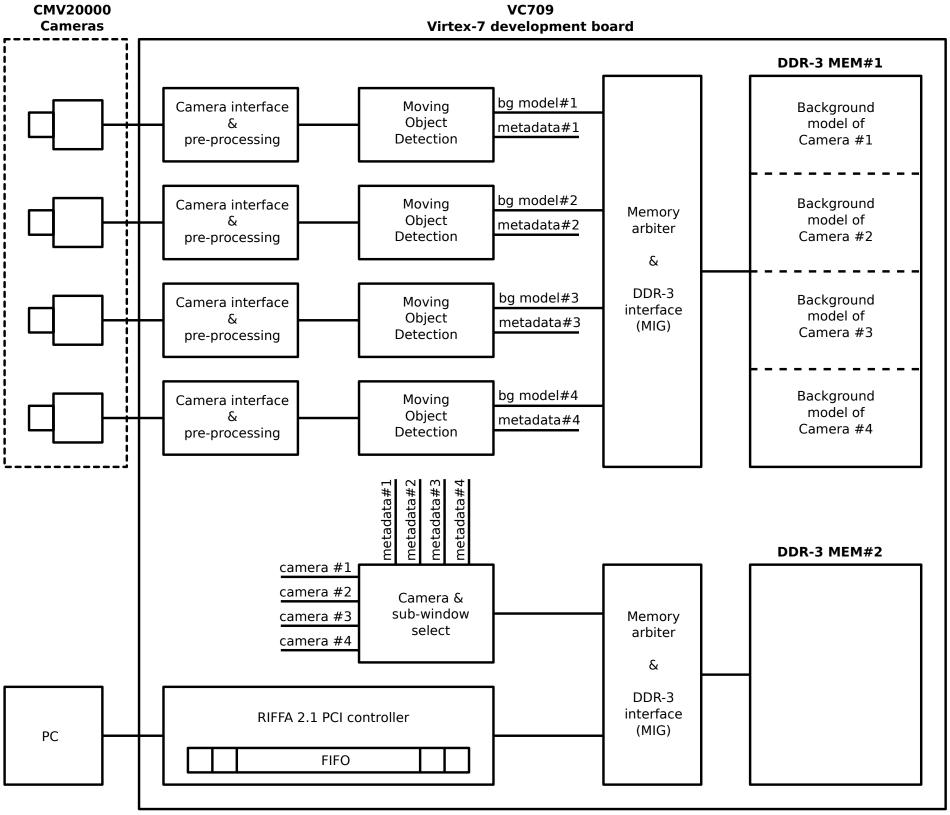

CMV20000
Cameras
VC709
Virtex-7 development board
DDR-3 MEM#1
Camera interface & pre-processing
Camera interface & pre-processing
Camera interface & pre-processing
Camera interface & pre-processing
Moving Object Detection
Moving Object Detection
Moving Object Detection
Moving Object Detection
bg model#1
metadata#1
bg model#2
metadata#2
bg model#3
metadata#3
bg model#4
metadata#4
Memory arbiter & DDR-3 interface (MIG)
Background model of Camera #1
Background model of Camera #2
Background model of Camera #3
Background model of Camera #4
metadata#1
metadata#2
metadata#3
metadata#4
DDR-3 MEM#2
camera #1
camera #2
camera #3
camera #4
Camera & sub-window select
Memory arbiter & DDR-3 interface (MIG)
PC
RIFFA 2.1 PCI controller
FIFO

Figure 4.21 – Smart camera concept applied to the GigaEye system

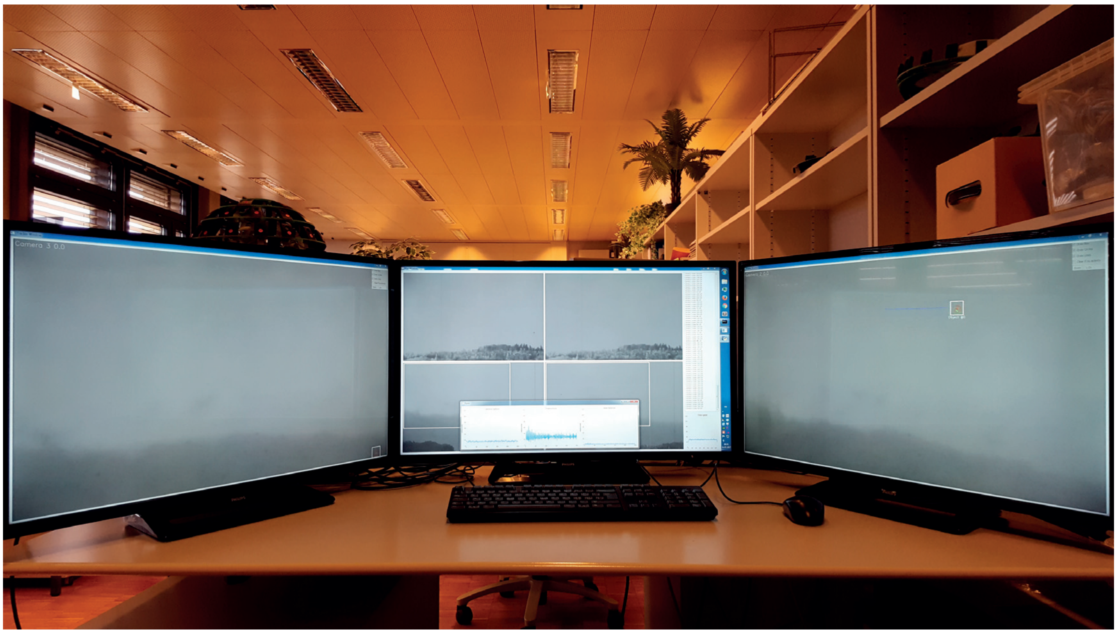
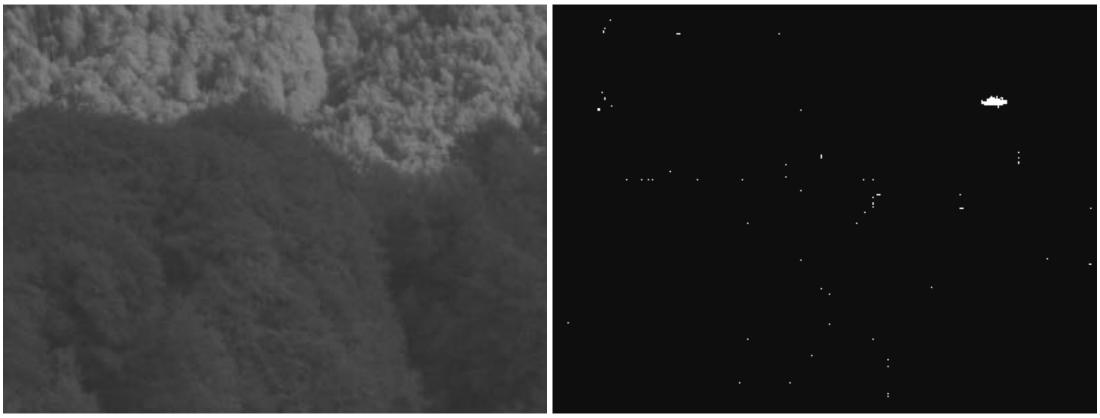


Figure 4.22 – Operation of the drone detection software for GigaEye platform

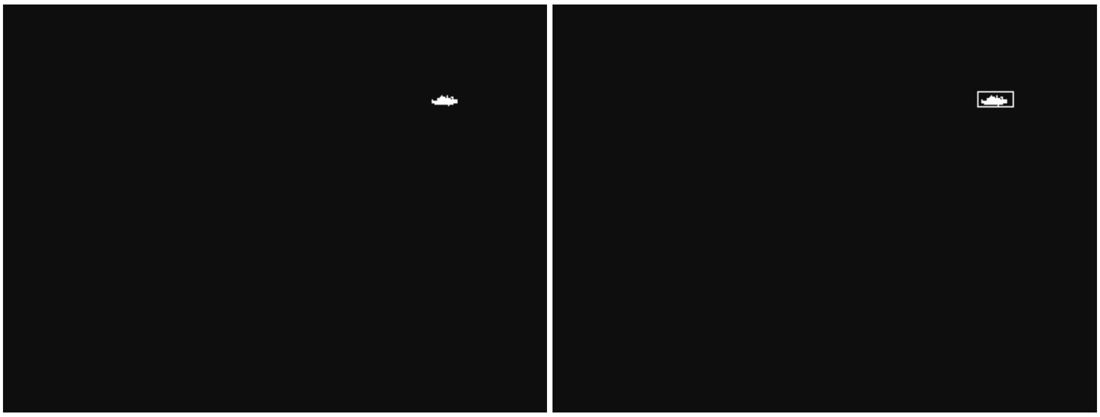
4.3.2 Result

The GigaEye-II system is demonstrated in drone detection days organized by armasuisse Science and Technology in 2016 and 2017, in Thun, Switzerland. In Figure 4.23, drone is at approximately 700 m distance from the system. Since there is no moving close object in the scene, a reliable background is computed. Figure 4.23b shows the difference the frame and the background model, and Figure 4.23c is the output of the morphological filter which removes all the noisy pixels. In Figure 4.23d connected component analysis result can be clearly seen that the drone is detected.



(a) Frame

(b) Background subtraction

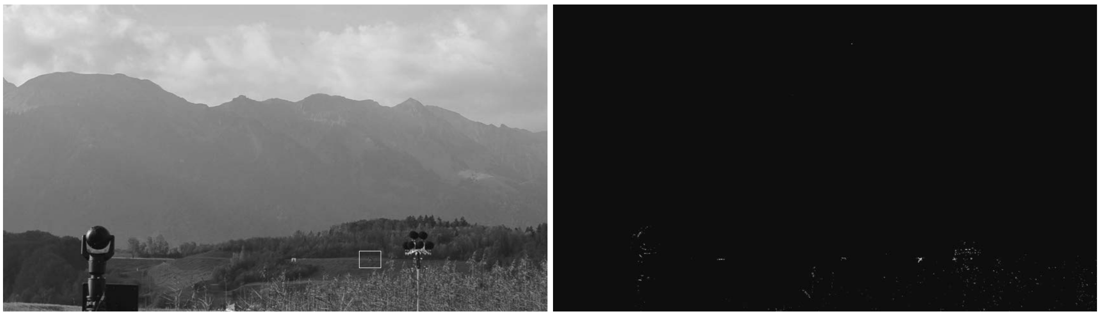


(c) Morphologic filter

(d) CCL

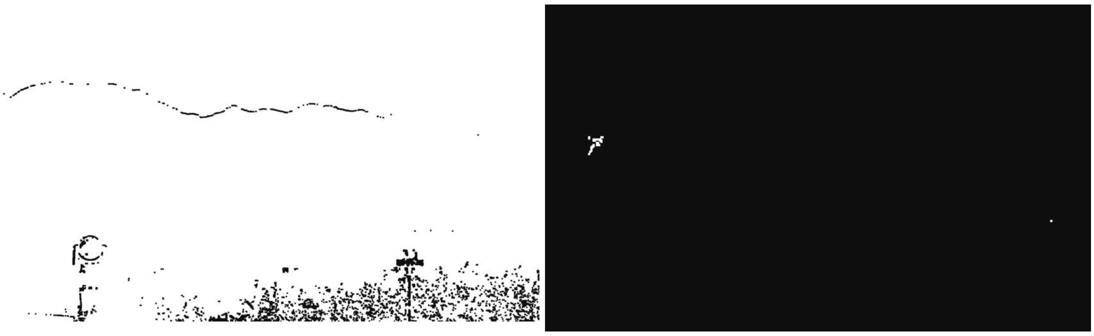
Figure 4.23 – Moving Object Detection stages

Another case is shown in 4.24. In this case, there are non-salient movements occurs caused by bushes and other detection systems. In the difference frame in Figure 4.24b, the noise and drone can be seen together. However this noise is masked by the reliability matrix shown in Figure 4.24c. Finally in Figure 4.24d, the target drone can be clearly seen. Although there is some corrosion with its shape, it is still recognizable. Since the recognition algorithm does not utilize this shape, instead it uses a non-compressed but cropped from the original frame, it does not have a side effect.



(a) Frame, drone is manually marked

(b) Difference and threshold



(c) Reliability

(d) Foreground output (zoom)

Figure 4.24 – Moving Object Detection stages

4.4 Wireless Smart Camera Networks

4.4.1 Introduction

In Section 4.3, we analyzed drone detection application with a gigapixel resolution system, where power is not a concern and high data transmission rates are acceptable. However, several tasks such as wide-area surveillance in highlands and wild-life monitoring brings power- and bandwidth-limitations. A widely used approach to long-range surveillance is using a high resolution pan-tilt-zoom (PTZ) camera with a high-focal length lens. It can be connected to a PC, and some computer vision methods can be used to detect the targets. However this solution is not cost-efficient, due to requirement of high-focal length lens and high resolution image sensor. Moreover, power, data cabling and maintenance add an extra cost and these reduce flexibility.

Instead, energy autonomous wireless smart cameras distributed in the target area as depicted in could provide an efficient solution. Each camera node has its own processing and energy harvesting unit and is designed as low-cost. As described in Section 3, our smart camera architecture is aimed at producing metadata at low-power and low-latency. Such featured camera is therefore suitable to the Wireless Visual Sensor Network (WVSN) concept. By incorporating this hardware into an embedded microprocessor and into analog-RF modules, a complete SoC solution can be used as a smart sensor node.

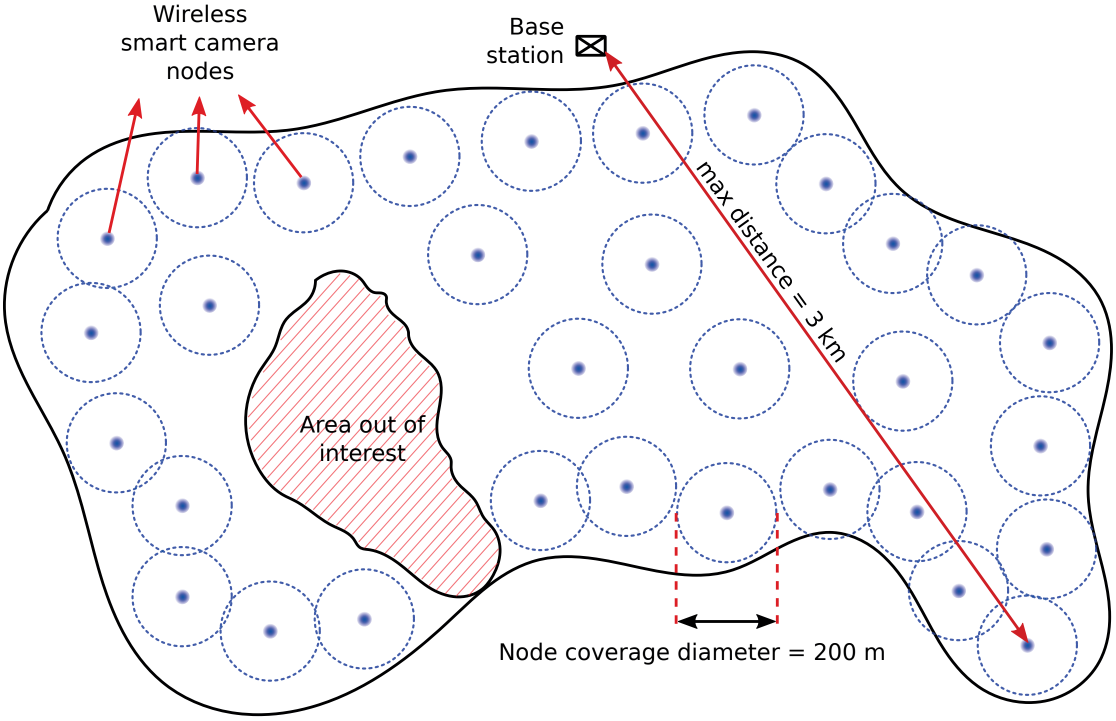


Figure 4.25 – Wide area surveillance with distributed camera nodes

Figure 4.25 illustrates our proposed distributed camera network where multiple cameras cover a wide zone. In this network, the area covered by each camera is reduced to decrease local power consumption at the expense of having multiple nodes dispersed over the land. Having omnidirectional imaging capability, each node covers a circular range in which the detection of an intrusion is possible. In this configuration, no camera node is placed on areas that are out of interest. Each node transmits the data wireless to the base station. To provide energy sustainability, energy harvesting is required directly at each camera node. An effective way to acquire enough energy from the environment would be to recover solar energy. This way, a direct DC voltage supply could be obtained. The base station is supposed to be powered by the mains, or an other available energy source.

In this part, we detail our proposition for a Wireless Smart Camera node which exploits our smart camera architecture. We have implemented the critical parts in hardware, targeting FPGA and ASIC. In addition, we show a full-system design concept, power analysis of the system to show feasibility of such an energy autonomous system.

4.4.2 Requirement Analysis

In this part, we present a detailed requirement analysis to determine the parameters of our proposed wireless smart camera system. Our goal is to send information from the local node to the central unit which is supposed to be at 1-3 km distance from the local node. Considering the distance and data rate requirement, sub-1GHz band is suitable for the data transmission.

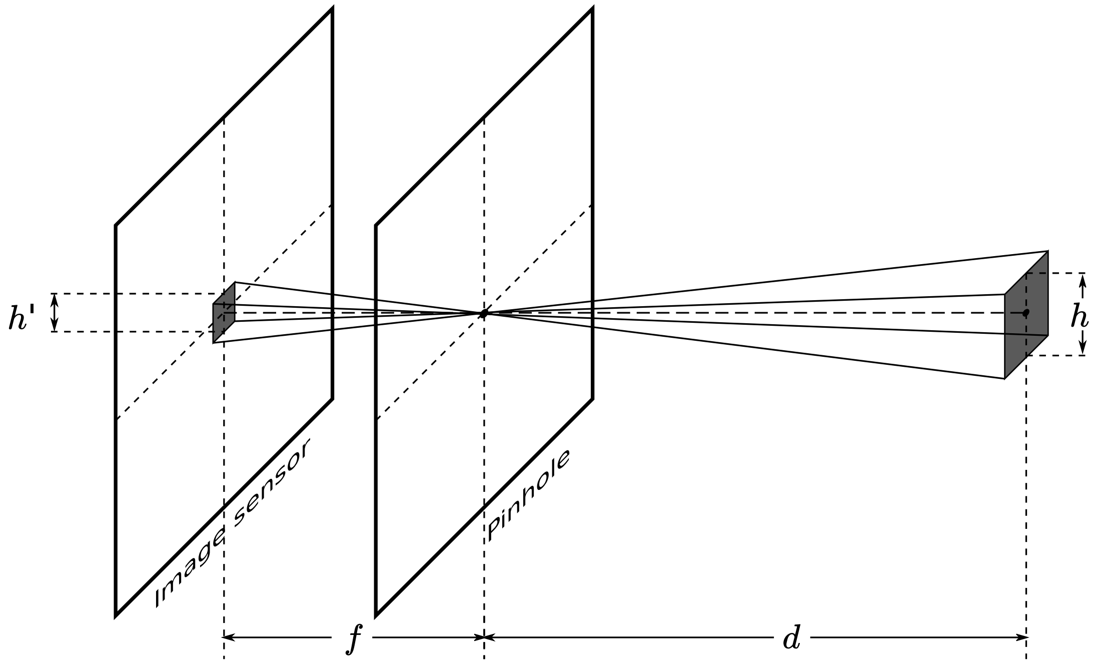


Figure 4.26 – Pin-hole camera model

Coverage of a node is defined as the field of view in which a camera can detect a target object, and it is an important parameter to determine the distribution of the cameras. This node coverage diameter is calculated using similarity of the triangles applied on the pin hole camera model illustrated in Figure 4.26. If the focal length of the lens is f, physical object height is h and projection height is h', distance of the target object from the camera, *i.e.* the node coverage radius, d is given in Equation 4.7:

$$d = f \times \frac{h_{im}}{h_{re}} \tag{4.7}$$

Taking into account the object detection algorithms, the object should be at least 12-pixels big in the image frame in order to be clearly identified. If a 2.25 mm lens and 1.8 mm height VGA sensor are used, radius of a node coverage can be calculated as follows:

$$d = 2.25\text{mm} \times \frac{2m}{\frac{12\text{px}}{480\text{px}} \times 1.8mm} = 100m \tag{4.8}$$

This means that a single camera node observes an area with 100 m radius. Consequently, a minimum 200 m distance between the camera nodes prevents processing overlapping information.

Moving object detection is the primary task operating continuously in the lowest digital energy level. As analyzed in Section 3.2, this task needs 18 bits per pixel. When a VGA sensor is used, $640 \times 480 = 300k$ pixel needs 4.5 Mb memory space, so the memory requirement of 6 cameras is 27 Mb for the moving object detection task.

During a day, the number of detected salient activities and the compression rate determine the bandwidth requirement. If we assume that one event occur per minute during 15 hours of operation and that stream is reduced by cropping and compressing to its 5% the data to be transferred per day is approximately 100 Mbits.

For 868 MHz carrier frequency, we can calculate the path loss by using Friis equation:

$$P_R = \frac{P_T G_T G_R \lambda^2}{(4\pi R)^2} \tag{4.9}$$

In this equation P_R, P_T, G_T and G_R stand for power and antenna gains at receiver and transmitter respectively, while λ is operation wavelength and R is the transmission distance. This equation can be converted into the logarithmic scale as in the following:

$$P_R(dB) = P_T(dB) + G_T(dB) + G_R(dB) - 20\log_{10} R - 20\log_{10} f + 20log_{10}\frac{c}{4\pi} \tag{4.10}$$

Here, the path loss is defined as given by:

$$\begin{aligned} L &= 20\log_{10} R + 20\log_{10} f - 20log_{10}\frac{c}{4\pi} \\ &= 20\log_{10} R + 20\log_{10} f - 147.55 \end{aligned} \tag{4.11}$$

Similarly to commercial wireless communication units in the market, we initially assume that output power is 10 dBm, transmitter and receiver antenna gains are 3 dBi as were given in Table 4.4. In order to transmit the data to the central node at 3 km distance, receiver sensitivity should be -105 dBm including 15 dB range for weather conditions.

Table 4.4 – Wireless power analysis

Frequency	868	MHz
Output power	10	dBm
TX antenna gain	3	dBi
RX antenna gain	3	dBi
Distance	3	km
Path loss	105.25	dBc
Power at RX	-89.25	dBm

The power management of the system is constituted of solar panels, battery charging circuits and DC-DC converters. Supposing a cylindrical architecture for the system with a radius $r = 10cm$, the available solar panel area is $\pi r^2 = 314cm^2$. It delivers an output voltage of 3 V with a maximum peak current of 200 mA. Its typical conversion efficiency is 15%. Harvested energy is proportional to solar irradiance, as well as area and efficiency of the solar panel. Solar irradiance in Lausanne during a year is shown in Table 4.5. Considering the worst case scenario, we can assume solar irradiance of $1kWh/m^2/$day for energy calculations. Accordingly, harvested energy per day in the worst case scenario is shown in Table 4.6.

Table 4.5 – Solar irradiance [$kWh/m^2/$day] on a horizontal surface in Lausanne [116]

Jan	Feb	Mar	Apr	May	Jun	Jul	Aug	Sep	Oct	Nov	Dec
1.39	2.16	3.33	4.24	5.17	5.90	5.97	5.08	3.74	2.31	1.43	1.08

Table 4.6 – Harvested energy per day in the worst case

Solar irradiance	1000	$Wh/m^2/$day
Solar panel surface area	0.03	m^2
Solar panel efficiency	15	%
Harvested energy per day	4.5	$Wh/$day

Considering a 300 cm^2 solar panel area, the total available energy was calculated in Table 4.6 as 4.5 Wh/day. The proposed case has an active area of 264.6 cm^2, which makes the total available energy equal to around 4 Wh/day.

Weather conditions varying a lot from one day to the other, the illumination on the panel changes consequently, too. The 3V CR123A Lithium-Ion battery, can be chosen in order to store solar energy and deliver it to the circuit in bad weather conditions. Having a capacity of 400mAh, the total energy is $3V \times 400mAh = 1.2Wh$. Several batteries may be placed in parallel to increase the total capacity with respect to the application. A battery charging circuit will be used in order to transfer the energy of the panel to the battery. The bq25505 boost charger circuit of Texas Instruments (TI) enables battery charging even for input voltages as low as 100mV. Moreover, it has a battery management block which could be very efficient in this particular case. With the maximum input voltage being 5.5V, this circuit is suitable for the chosen 3V solar panel. Each bq25505 circuit could tolerate a maximum input power of 510mW. Thus, 5 bq25505 circuits is necessary to cover all the solar panels.

These circuits are always on, constantly charging the battery. The data sheet shows a charging efficiency of 90% approximately, with a little drop to 80% when the input power goes higher than 100mA. Since each solar cell may deliver a peak current of 200mA, the conversion efficiency is taken as 80% for the calculus. Equation 4.12 shows the energy lost E_{LOST} during the harvesting with respect to the solar energy E_{IN} and conversion efficiency η:

$$E_{LOST} = E_{IN} \times (1-\eta) = 4 \times (1-0.8) = 0.8Wh/\text{day} \tag{4.12}$$

The passive energy is negligible compared to the conversion loss in this case.

Image sensor is another important power consumer. The typical current consumption of our camera is detailed as 40mA in active mode. The quiescent current during sleep mode is 0.1 mA. Since only one camera is active at a given time, and assuming 15 operating hours per day with 60 mW power consumption, we can estimate 900 mWh energy budget is needed for image sensors.

The total power consumption is calculated as 56.5mW, which corresponds to an energy of 1.426 Wh/day. With the solar panel providing 4 Wh/day, the system is operational with an energy excess of more than 2 Wh/day. This analysis shows that such a system is feasible with respect to the system requirements. Since there is an available extra power budget, the system could be even more improved.

4.4.3 System overview

As previously mentioned, our primary objective is to observe an area with minimum cost and maintenance. In this scope, extending angle of view to 360 degree increase the coverage of a single node. Therefore, this omnidirectional imaging capability reduces the number of the nodes. Although this can be implemented using a single rotating camera, having such mechanical system increases the power consumption and requires more maintenance. Hence, we propose to cover 360 degree horizontal angle of view with low cost multiple cameras by achieving digital rotation. Since commercial camera units have typically around 60 degree horizontal angle of view, 6 cameras are sufficient for 360 degree viewing.

A simplified block diagram of the proposed multi-camera system is shown in Figure 4.27.

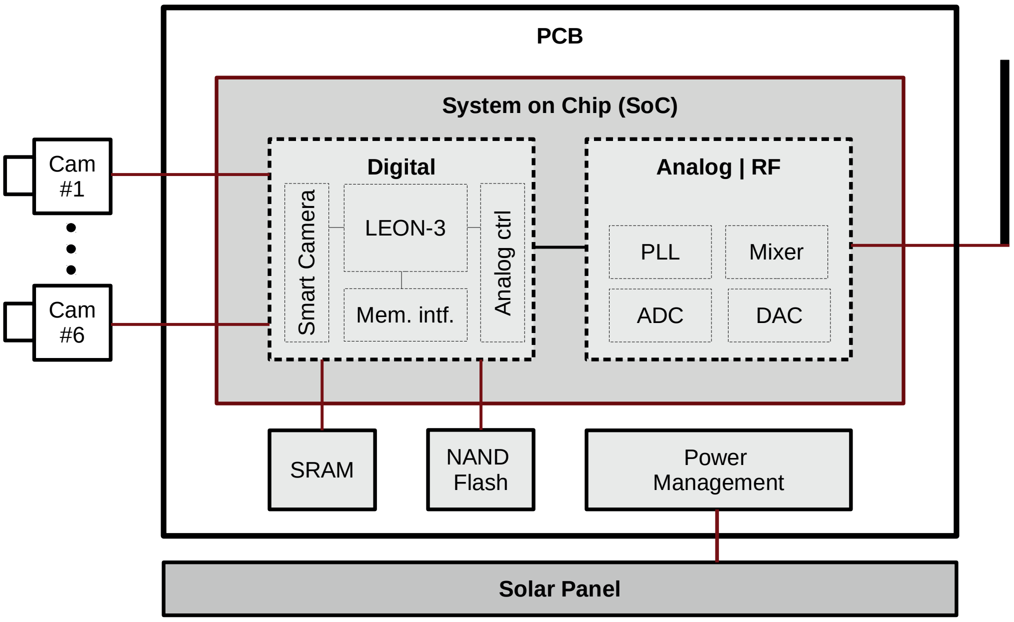


Figure 4.27 – WVSN node block diagram

Digital and analog blocks are placed in the same chip to build the SoC. The control of the system is performed by the Leon-3 core which is an open-source 32-bit RISC processor. The other digital blocks including the smart camera, memory interface and analog control blocks are connected to the bus system of the Leon-3 processor.

The SoC needs external connections for its functionality. The smart camera sub-block is connected to the cameras. Run-time and long-term data storage is provided by external SRAM and NAND Flash memories, respectively. Analog module has PLL, ADC, DAC and Mixer blocks to support different analog sensors, in addition to Sub-1GHz wireless communication.

In this first prototype, we aim designing energy harvesting circuitry with on-the-shelf discrete components. A photovoltaic panel recovers solar energy and provide a 3V constant DC voltage. DC-DC converters on the PCB generates the voltage levels necessary for the SoC and the other components.

If the smart camera detects a moving target, it either sends an image patch to the base station, or it stores it into the NAND Flash depending on the application.

Six cameras capture one at a time 1 frame per second and transmit the data to a low power FPGA. An SRAM memory stores a background image from each camera. With the weather conditions varying all the day, the background images are updated periodically. The FPGA processes data coming from the cameras and detects if any intrusion occurred. In the latter case, relevant data is sent to the wireless transceiver. This one creates a link with the base station and send the data wirelessly. An omnidirectional antenna is used for RF transmissions. Even if a large ratio of the electromagnetic power is lost through unused directions, the omnidirectional antenna brings some ease to the user to place the system on the high land. Indeed, with an omnidirectional antenna, the user does not have to worry about the direction wherever the system is positioned.

Considering cost, power and availability in the market TOSHIBA TCM8230MD VGA camera can be chosen for the proposed system. The sensor array has 640 × 480 pixels and has already its lens incorporated.

An energy harvesting circuit is implemented in order to store the solar energy into a battery. Giving that the solar irradiance is fully variable, the output power of the panel may present variations that could cause the circuits to malfunction. Thus, a battery could filter these fluctuations and ensure proper operation of the system.

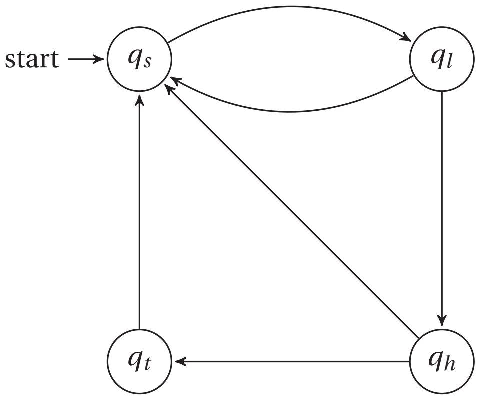


Figure 4.28 – FSM for operation of the WVSN

The principle of operation of the smart camera functionality in the node is shown in Figure 4.28. Firstly, one of the cameras is activated to drive the camera connections while the other cameras are disabled in initial state q_s . Then it switches to the state q_l which is the low power state where moving object detection block handles the incoming camera data, and performs its operation by accessing the background model stored in the SRAM. If moving objects are detected in this analysis, feature extraction module is activated for the next frame of the same camera in the state q_h. FREAK features can be extracted and the target object can be detected by comparing the extracted FREAK features with the pre-stored object features. Finally, if the probability of the moving object to match with the pre-defined targets exceeds a threshold, FSM goes into state q_t, which sends the related part of the image wirelessly and needs the most power.

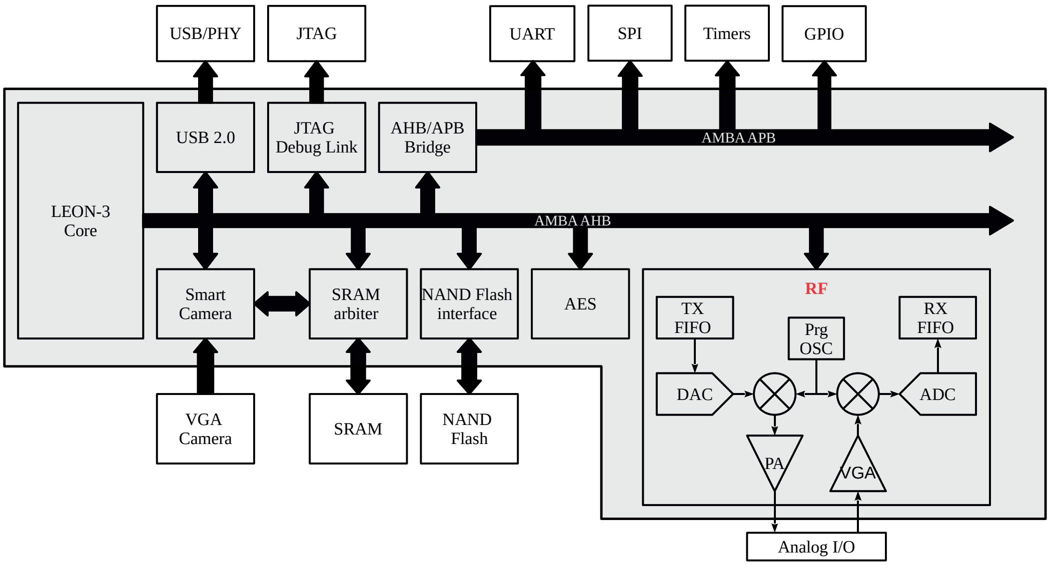


Figure 4.29 – Block diagram of the processing chip

4.4.4 Electronics Design

Block diagram of the processing chip of the system is shown in Figure 4.29. The system consists of an open source 32-bit Leon-3 RISC processor with Sparc architecture [117]. Leon uses the ARM Advanced Microcontroller Bus Architecture (AMBA), with the Advanced High Performance Bus (AHB), and a slower peripheral bus (APB). Connection between the two buses is provided by AHB/APB bridge.

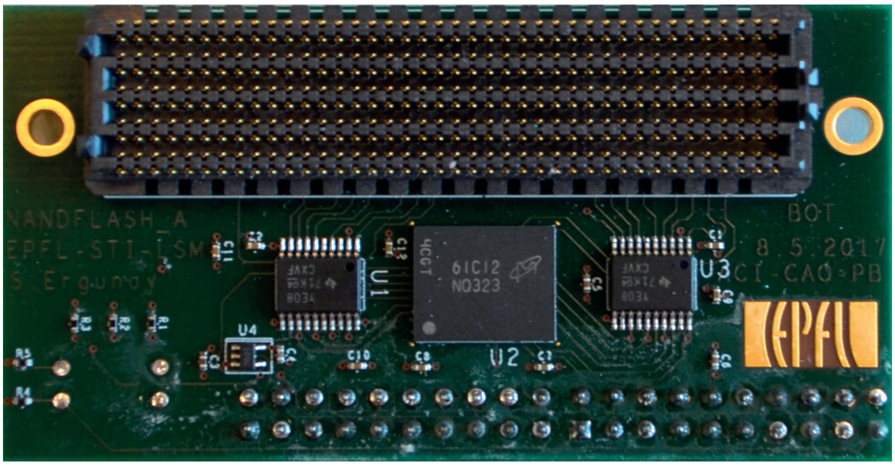


Figure 4.30 – NAND Flash interface board equipped with FMC connector

In addition to data transmission with the wireless communication, data storage is also enabled in the system with a NAND Flash memory. This memory is controlled by a NAND Flash interface connected to the AMBA AHB. In order to test the NAND Flash controller, we designed the PCB shown in Figure 4.30, with the 16 GB NAND Flash (model name: MT29F16G08ADACAH4). The connectivity of the board is provided by an FMC connector, and a dip header. The board can be connected to a FPGA board, and dip header is pin compatible with Arduino Uno rev3.

Both FPGA and MCU boards can be connected separately or simultaneously. Access to the NAND Flash is controlled by a NOR gate connected to the enable pin. At a given time, only one unit can access the NAND Flash. NAND Flash is connected to bus, unless the both units send 0. In that case, NAND is unconnected to the bus with high-Z, and MCU and FPGA can communicate via the bus.

In this design, we integrated a low-power and wide-range Analog to Digital Converter (ADC) and Variable Gain Amplifier (VGA) presented in [118]. These ADC and VGA are designed for from 83.3 kS/s to 85.7 MS/s sampling range. Thus, it is suitable to use it in different sensor applications as well as sub-1GHz communication by incorporating a PLL to generate the carrier frequency and a mixer. In case of the PLL is activated, the system can communicate in sub-1GHz band, by injecting the carrier signal to the analog I/O via mixer. When the PLL is disabled, the analog sub-system can be used to capture the signals in this range.

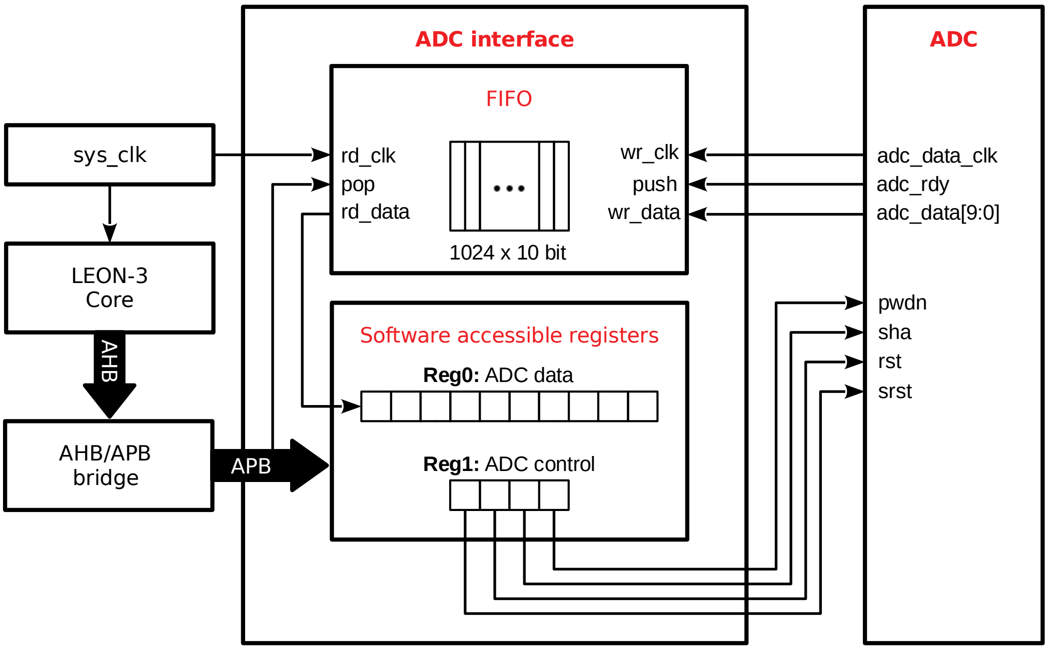


Figure 4.31 – Block diagram of ADC interface

Sensor data or incoming data converted by ADC is transferred to the system via ADC interface as shown in Figure 4.31. Leon writes to the software accessible registers to control the ADC. 4 registers are connected to power down (pwdn), sample-hold adjustment (sha), reset (rst) and SAR reset (srst) inputs of the ADC. On the other hand, outputs of the ADC are connected to the FIFO pins directly. ADC sends the 10-bit data with the ready and clock signals, which are connected to write clock, push and data ports of FIFO respectively. Thus, ADC data is pushed to the 1024-depth FIFO to prevent the data loss. Then processor or DMA unit pops the ADC data via the software accessible register, reg0.

In order to test the design, we incorporate different blocks and taped out them in 65-nm TSMC technology. Chip layout is presented in Figure 4.32. On the left, an ADC block and ADC+VGA blocks are separately placed in order to get some measurements. Moreover, on the right ADC is integrated to digital core, which has memory, keypoint detection hardware, NAND Flash controller and ADC controller, all controller by LEON-3. The area of the chip is 3 mm × 2 mm = 6 mm^2. However, since some components are replicated for test purposes, the actual area is expected to be 3 mm^2.

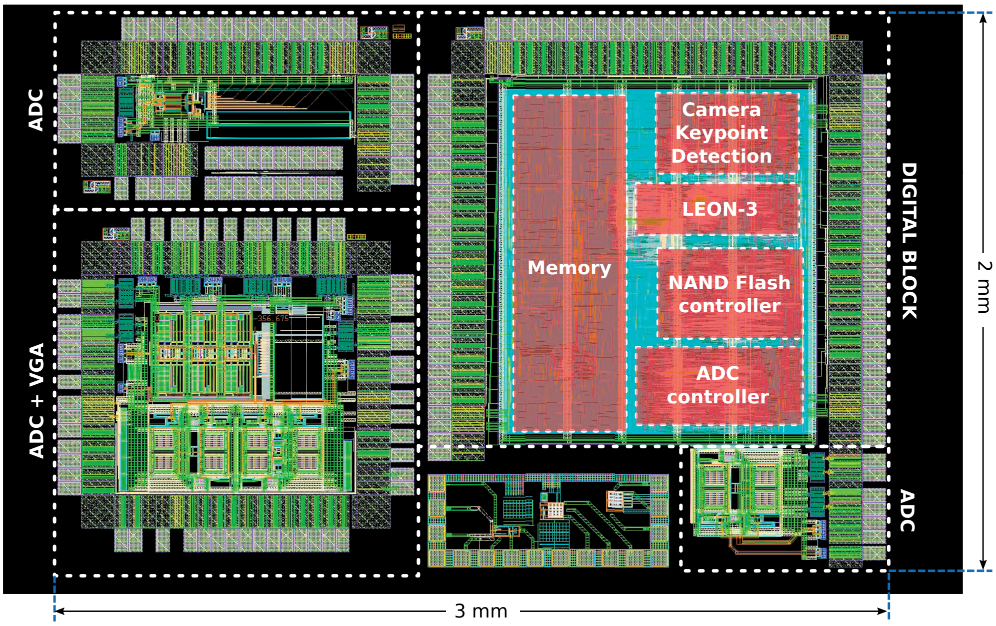


Figure 4.32 – Tapeout in 65-nm TSMC technology

Mechanical design

Along with the electronic design, a first mechanical prototype is designed for the wireless camera node. A 3D view of the disassembled system is shown in Figure 4.33 with all of its components. In order to get maximum solar energy, the panel is placed on the very top of the system. The omnidirectional RF antenna is placed in the middle of the panel to minimize RF losses. This positioning will generate a small shadow on the panel, reducing its efficiency. However, the ratio between the shadow and the illuminated area would be as small as 6% in the worst case. The power drop due to eventual shadows on the panel will not be high enough to make the system malfunction. The panel stands on the mechanical top plate, in which a hole gives access to the panel cables. The central part contains the 8 cameras at equal angles of 45° from the center. The main-board containing all the circuitry is also mounted in this central part.

The bottom part of this design contains the batteries. A power connector connects the batteries to the solar panel and the main board through a hole. A long post suited to be fixed in the ground supports the full system. A side view and an assembled view of the system are shown in Figure 4.33. The supporting post has been designed 1m high and the system diameter is hold 20 cm wide. Depending on the application and the area to survey, those dimensions can be adapted.

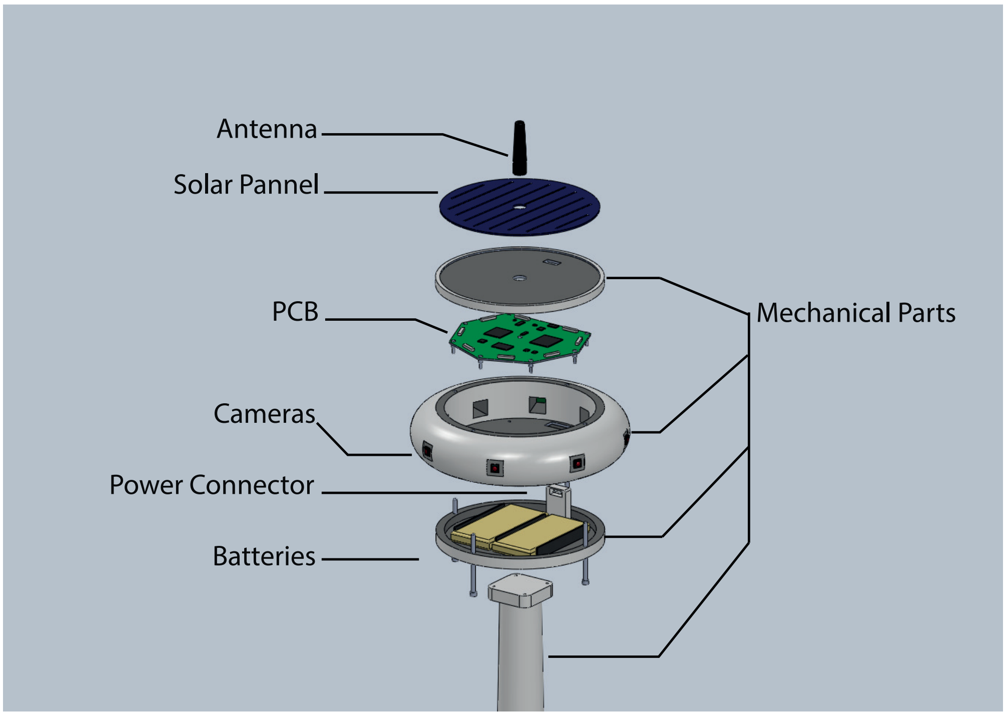


Figure 4.33 – 3D view of the energy autonomous camera system

4.4.5 Conclusion

In this section we gave design specifications of a solar powered wireless smart camera network and performed power budget analysis on the system considering requirements. In order to reduce power consumption of the sensor nodes, we proposed local processing of the captured images. With our strategy, we show that the proposed system can fully operate as energy-autonomous. We also designed mechanical structure of the nodes, in addition to the electronic units. In the current design, energy harvesting circuitry designed with discrete components. In case of integration to the SoC, more efficient systems can be reached.

5 Conclusion

In this thesis, we presented a smart camera architecture for multiple and wireless smart camera applications. The processing blocks of the proposed architecture were described in detail, and application examples were given. In this Chapter, we summarize our work, clarify our contributions and give future research directions.

5.1 Architecture

Proposed architecture is based on generating metadata with on-the-fly pixel operations. In other words, relevant information about the content of the frame is generated during the pixel transfer from the sensor to the processing unit, and completed with the end of frame. As such, we presented processing blocks, namely moving object detection, feature extraction and cellular neural network, each requiring different amount of energy. Adapting the energy levels within one system depending on the application provides resource efficiency while still keeping the scene understanding at a reasonable accuracy.

In particular, the first block performs a low-complexity and low-power moving object detection algorithm. This block requires less memory comparing to its alternatives while being resistant to non-salient motion. Therefore, it is suitable to be operated continuously, and used to trigger a further analysis when a salient motion is detected.

The second block is feature extraction block which combines FAST corner detection and FREAK binary description algorithms. These algorithms are implemented in a hardware efficient way, which brings a power and timing efficiency.

As the third block, we proposed a CNN-based architecture, Hybrid Processor Population. Differently from CNN, it incorporates inhibitory and excitatory cells inspiring from biological neural networks. Due to its configurability with identity matrix, many different networks either regular or random distributions can be obtained. Accordingly, we gave some network examples of such Hybrid Processor Population that perform LBD computations.

5.2 Applications

Technological developments and reduced costs of the smart camera components, *i.e.*, image sensor, processing device and memory, leads to expansion of smart camera applications. Accordingly, we presented four applications in this work.

The typical application of compound cameras is omnidirectional imaging. Real-time omnidirectional imaging applications needs high bandwidth. Therefore, we proposed a high bandwidth hybrid architecture based on tree network topology. Based on this, we presented its PCB which has a high performance FPGA, QDR-II memories and supports 16 cameras.

In the next application, we developed a vision-based drone surveillance system with 16 of high resolution cameras. High resolution image sensors increases the detection range, but they also brings bandwidth limitations. We examined a GPU-based system, then proposed a background subtraction-based method to deal with the bandwidth problems. In this application, we benefit from the metadata which gives the coordinate and size of the moving objects.

Polarimetric imaging provides useful information regarding image/scene content such as material properties, shape and roughness of the object. We used polarimetry in object detection tasks, particularly to distinguish man-made vehicles from natural backgrounds, leveraging the deep learning architectures. Since there is no public dataset of polarimetric images during this project, we firstly collected data to train the deep networks. We showed that polarimetry can enhance the performance of the object detection systems, and proposed a low-cost multi-camera polarimetry device.

The final application is the use of smart camera as a Wireless Smart Camera Network node. Smart camera module is integrated into a 32-bit RISC processor-based embedded system with sub-1GHz wireless communication and NAND Flash modules. Profiting from the proposed smart camera architecture with adaptable energy consumption, it has processing units that can function properly in power-critical applications. Moreover, the proposed system was partially taped out in TSMC 65-nm technology.

5.3 Future prospects

Today, multiple and wireless vision systems deal with challenges such as power, bandwidth, memory and processing capacity. In order to solve them, there is not only a single research direction, but different approaches in different design levels are needed. In system-level, one approach could be algorithm optimization to reduce memory access and computational cost. In transistor-level, low-power image sensor design, focal-plane image processing, near-threshold operations, approximate computation, Energy-Quality scalability, could be a solution to handle resource limitations. In addition to these, data compression can be addressed to effectively operate latest emerging deep learning schemes that are also highly resource-demanding. A number of other solutions can be proposed in order to create more and more efficient smart camera designs.

On the other hand, more classical methods with off-the-shelf components and low-cost processing devices such as Raspberry Pi, some FPGA and GPU boards, incorporates open source libraries like OpenCV, which leads to flexible designs reducing development time, hence development cost. Although they lack of power efficiency compared to ASIC methods, some amount of energy could be saved with system-algorithm-level optimization. Especially in low-volume manufacturing, these solutions fit well to the problem.

There are ongoing researches to augment intelligence in smart cameras with low-power and low-cost embedded systems designs. Obviously, this has the potential to bring privacy concerns as much as smart cameras become prevalent in daily life. This being already controversial issue with the latest breakthroughs in Artifical Intelligence, should be considered seriously and studies regarding data encryption in smart cameras should be addressed as a next future direction.

Bibliography

[1] A. N. Belbachir and P. M. Göbel, “Smart Cameras: a Historical Evolution”, in *Smart Cameras*, A. N. Belbachir, Ed., Boston, MA: Springer US, 2010, pp. 3–17.

[2] The Editors of Encyclopaedia Britannica, *Nicéphore niépce — encyclopædia britannica*, https://www.britannica.com/biography/Nicephore-Niepce, [Online; accessed 12-August-2018], 2018.

[3] Y. Shi and F. D. Real, “Smart Cameras: Fundamentals and Classification”, in *Smart Cameras*, A. N. Belbachir, Ed., Boston, MA: Springer US, 2010, pp. 19–34.

[4] S. Hengstler, D. Prashanth, S. Fong, and H. Aghajan, “MeshEye: a Hybrid-Resolution Smart Camera Mote for Applications in Distributed Intelligent Surveillance”, in *2007 6th International Symposium on Information Processing in Sensor Networks*, Apr. 2007, pp. 360–369.

[5] B. Wilburn, N. Joshi, V. Vaish, E.-V. Talvala, E. Antunez, A. Barth, A. Adams, M. Horowitz, and M. Levoy, “High Performance Imaging Using Large Camera Arrays”, in *ACM SIGGRAPH 2005 Papers*, ser. SIGGRAPH ’05, New York, NY, USA: ACM, 2005, pp. 765–776.

[6] D. Anguelov, C. Dulong, D. Filip, C. Frueh, S. Lafon, R. Lyon, A. Ogale, L. Vincent, and J. Weaver, “Google Street View: Capturing the World at Street Level”, *Computer*, vol. 43, no. 6, pp. 32–38, Jun. 2010.

[7] Y. Yagi, “Omnidirectional Sensing and Its Applications”, *IEEE Trans Inf and Syst E*, vol. E82-D, Mar. 1999.

[8] K. Kirschfeld, “The Resolution of Lens and Compound Eyes”, in *Neural Principles in Vision*, ser. Proceedings in Life Sciences, Springer, Berlin, Heidelberg, 1976, pp. 354–370.

[9] Y. M. Song, Y. Xie, V. Malyarchuk, J. Xiao, I. Jung, K.-J. Choi, Z. Liu, H. Park, C. Lu, R.-H. Kim, R. Li, K. B. Crozier, Y. Huang, and J. A. Rogers, “Digital cameras with designs inspired by the arthropod eye”, *Nature*, vol. 497, no. 7447, pp. 95–99, May 2013.

[10] S. K. Nayar, “Catadioptric omnidirectional camera”, in *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, Jun. 1997, pp. 482–488.

[11] J. Foote and D. Kimber, “FlyCam: practical panoramic video and automatic camera control”, in *2000 IEEE International Conference on Multimedia and Expo. ICME2000. Proceedings. Latest Advances in the Fast Changing World of Multimedia (Cat. No.00TH8532)*, vol. 3, 2000, 1419–1422 vol.3.

[12] O. Schreer, I. Feldmann, C. Weissig, P. Kauff, and R. Schafer, “Ultrahigh-Resolution Panoramic Imaging for Format-Agnostic Video Production”, *Proceedings of the IEEE*, vol. 101, no. 1, pp. 99–114, Jan. 2013.

[13] D. J. Brady, M. E. Gehm, R. A. Stack, D. L. Marks, D. S. Kittle, D. R. Golish, E. M. Vera, and S. D. Feller, “Multiscale gigapixel photography”, *Nature*, vol. 486, no. 7403, pp. 386–389, Jun. 2012.

[14] Y. Xu, Q. Zhou, L. Gong, M. Zhu, X. Ding, and R. K. F. Teng, “High-Speed Simultaneous Image Distortion Correction Transformations for a Multicamera Cylindrical Panorama Real-time Video System Using FPGA”, *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 24, no. 6, pp. 1061–1069, Jun. 2014.

[15] H. Afshari, A. Akin, V. Popovic, A. Schmid, and Y. Leblebici, “Real-Time FPGA Implementation of Linear Blending Vision Reconstruction Algorithm Using a Spherical Light Field Camera”, in *2012 IEEE Workshop on Signal Processing Systems*, Oct. 2012, pp. 49–54.

[16] V. Popovic, K. Seyid, A. Akin, Ö. Cogal, H. Afshari, A. Schmid, and Y. Leblebici, “Image Blending in a High Frame Rate FPGA-based Multi-Camera System”, *Journal of Signal Processing Systems*, vol. 76, no. 2, pp. 169–184, Aug. 2014.

[17] K. Seyid, V. Popovic, O. Cogal, A. Akin, H. Afshari, A. Schmid, and Y. Leblebici, “A Real-Time Multiaperture Omnidirectional Visual Sensor Based on an Interconnected Network of Smart Cameras”, *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 25, no. 2, pp. 314–324, Feb. 2015.

[18] O. Cogal, A. Akin, K. Seyid, V. Popovic, A. Schmid, B. Ott, P. Wellig, and Y. Leblebici, “A new omni-directional multi-camera system for high resolution surveillance”, in *Mobile Multimedia/Image Processing, Security, and Applications 2014*, vol. 9120, International Society for Optics and Photonics, May 2014, 91200N.

[19] L. Gaemperle, K. Seyid, V. Popovic, and Y. Leblebici, “An Immersive Telepresence System Using a Real-Time Omnidirectional Camera and a Virtual Reality Head-Mounted Display”, in *2014 IEEE International Symposium on Multimedia*, Dec. 2014, pp. 175–178.

[20] O. Cogal and Y. Leblebici, “An Insect Eye Inspired Miniaturized Multi-Camera System for Endoscopic Imaging”, *IEEE Transactions on Biomedical Circuits and Systems*, vol. 11, no. 1, pp. 212–224, Feb. 2017.

[21] A. Rowe, C. Rosenberg, and I. Nourbakhsh, “A low cost embedded color vision system”, in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, vol. 1, Sep. 2002, 208–213 vol.1.

[22] 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”, in *Proceedings of the 3rd International Conference on Embedded Networked Sensor Systems*, ser. SenSys ’05, New York, NY, USA: ACM, 2005, pp. 192–204.

[23] P. Kulkarni, D. Ganesan, P. Shenoy, and Q. Lu, “SensEye: a Multi-tier Camera Sensor Network”, in *Proceedings of the 13th Annual ACM International Conference on Multimedia*, ser. MULTIMEDIA ’05, New York, NY, USA: ACM, 2005, pp. 229–238.

[24] P. Chen, P. Ahammad, C. Boyer, S.-I. Huang, L. Lin, E. Lobaton, M. Meingast, S. Oh, S. Wang, P. Yan, A. Y. Yang, C. Yeo, L.-C. Chang, J. D. Tygar, and S. S. Sastry, “CITRIC: a low-bandwidth wireless camera network platform”, in *2008 Second ACM/IEEE International Conference on Distributed Smart Cameras*, Sep. 2008, pp. 1–10.

[25] K. Abas, C. Porto, and K. Obraczka, “Wireless Smart Camera Networks for the Surveillance of Public Spaces”, *Computer*, vol. 47, no. 5, pp. 37–44, May 2014.

[26] Aseem Agarwala, *Automatic photography with google clips*, https://ai.googleblog.com/2018/05/automatic-photography-with-google-clips.html, [Online; accessed 15-September-2018], 2018.

[27] P. Chalimbaud and F. Berry, “Embedded Active Vision System Based on an FPGA Architecture”, *EURASIP J. Embedded Syst.*, vol. 2007, no. 1, pp. 26–26, Jan. 2007.

[28] M. Imran, K. Shahzad, N. Ahmad, M. O’Nils, N. Lawal, and B. Oelmann, “Energy-Efficient SRAM FPGA-Based Wireless Vision Sensor Node: SENTIOF-CAM”, *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 24, no. 12, pp. 2132–2143, Dec. 2014.

[29] J. Fernandez-Berni, R. Carmona-Galan, and L. Carranza-Gonzalez, “FLIP-q: a QCIF Resolution Focal-Plane Array for Low-Power Image Processing”, *IEEE Journal of Solid-State Circuits*, vol. 46, no. 3, pp. 669–680, Mar. 2011.

[30] J. Fernández-Berni, R. Carmona-Galán, G. Liñán-Cembrano, Á. Zarándy, and Á. Rodríguez-Vázquez, “Wi-FLIP: a wireless smart camera based on a focal-plane low-power image processor”, in *2011 Fifth ACM/IEEE International Conference on Distributed Smart Cameras*, Aug. 2011, pp. 1–6.

[31] S. Battiato, A. R. Bruna, G. Messina, and G. Puglisi, *Image Processing for Embedded Devices.* Bentham Science Publishers, Nov. 2010, Google-Books-ID: K5aOhnvGJToC.

[32] R. Carmona-Galán, Á. Zarándy, C. Rekeczky, P. Földesy, A. Rodríguez-Pérez, C. Domínguez-Matas, J. Fernández-Berni, G. Liñán-Cembrano, B. Pérez-Verdú, Z. Kárász, M. Suárez-Cambre, V. Brea-Sánchez, T. Roska, and Á. Rodríguez-Vázquez, “A hierarchical vision processing architecture oriented to 3d integration of smart camera chips”, *Journal of Systems Architecture*, Smart Camera Architecture, vol. 59, no. 10, Part A, pp. 908–919, Nov. 2013.

[33] R. G. Dreslinski, M. Wieckowski, D. Blaauw, D. Sylvester, and T. Mudge, "Near-Threshold Computing: Reclaiming Moore's Law Through Energy Efficient Integrated Circuits", *Proceedings of the IEEE*, vol. 98, no. 2, pp. 253–266, Feb. 2010.

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

[35] M. Rusci, D. Rossi, E. Farella, and L. Benini, "A Sub-mW IoT-Endnode for Always-On Visual Monitoring and Smart Triggering", *IEEE Internet of Things Journal*, vol. 4, no. 5, pp. 1284–1295, Oct. 2017.

[36] J. Koomey, S. Berard, M. Sanchez, and H. Wong, "Implications of Historical Trends in the Electrical Efficiency of Computing", *IEEE Annals of the History of Computing*, vol. 33, no. 3, pp. 46–54, Mar. 2011.

[37] K. Ahmed and K. Schuegraf, "Transistor wars", *IEEE Spectrum*, vol. 48, no. 11, pp. 50–66, Nov. 2011.

[38] M. Alioto, "Energy-quality scalable adaptive VLSI circuits and systems beyond approximate computing", in *Design, Automation Test in Europe Conference Exhibition (DATE), 2017*, Mar. 2017, pp. 127–132.

[39] A. B. Alvarez, G. Ponnusamy, and M. Alioto, "EQSCALE: Energy-quality scalable feature extraction engine for Sub-mW real-time video processing with 0.55 mm2 area in 40nm CMOS", in *2017 IEEE Asian Solid-State Circuits Conference (A-SSCC)*, Nov. 2017, pp. 241–244.

[40] C. Wren, A. Azarbayejani, T. Darrell, and A. Pentland, "Pfinder: real-time tracking of the human body", in *Proceedings of the Second International Conference on Automatic Face and Gesture Recognition*, Oct. 1996, pp. 51–56.

[41] C. Stauffer and W. E. L. Grimson, "Adaptive background mixture models for real-time tracking", in *Proceedings. 1999 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (Cat. No PR00149)*, vol. 2, Jun. 1999, 246–252 Vol. 2.

[42] M. Genovese and E. Napoli, "ASIC and FPGA Implementation of the Gaussian Mixture Model Algorithm for Real-Time Segmentation of High Definition Video", *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 22, no. 3, pp. 537–547, Mar. 2014.

[43] I. Haritaoglu, D. Harwood, and L. Davis, "W4: Who? When? Where? What? a Real Time System for Detecting and Tracking People", *Third Face and Gesture Recognition Conference*, Dec. 1997.

[44] S. S. Sengar and S. Mukhopadhyay, "Moving object detection based on frame difference and w4", *Signal, Image and Video Processing*, vol. 11, no. 7, pp. 1357–1364, Oct. 2017.

[45] M. U. K. Khan, A. Khan, and C. M. Kyung, "EBSCam: Background Subtraction for Ubiquitous Computing", *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 25, no. 1, pp. 35–47, Jan. 2017.

[46] M. Casares and S. Velipasalar, “Light-weight salient foreground detection for embedded smart cameras”, in *2008 Second ACM/IEEE International Conference on Distributed Smart Cameras*, Sep. 2008, pp. 1–7.

[47] J. Hoshen and R. Kopelman, “Percolation and cluster distribution. i. Cluster multiple labeling technique and critical concentration algorithm”, *Physical Review B*, vol. 14, no. 8, pp. 3438–3445, Oct. 1976.

[48] J. Trein, A. T. Schwarzbacher, and B. Hoppe, “FPGA Implementation of a Single Pass Real-Time Blob Analysis Using Run Length Encoding”, p. 7, 2008.

[49] R. Acevedo-Avila, M. Gonzalez-Mendoza, and A. Garcia-Garcia, “A Linked List-Based Algorithm for Blob Detection on Embedded Vision-Based Sensors”, *Sensors (Basel, Switzerland)*, vol. 16, no. 6, May 2016.

[50] M. Klaiber, L. Rockstroh, Z. Wang, Y. Baroud, and S. Simon, “A memory-efficient parallel single pass architecture for connected component labeling of streamed images”, in *2012 International Conference on Field-Programmable Technology*, Dec. 2012, pp. 159–165.

[51] M. J. Klaiber, D. G. Bailey, Y. O. Baroud, and S. Simon, “A Resource-Efficient Hardware Architecture for Connected Component Analysis”, *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 26, no. 7, pp. 1334–1349, Jul. 2016.

[52] K. Kim, T. H. Chalidabhongse, D. Harwood, and L. Davis, “Real-time foreground–background segmentation using codebook model”, *Real-Time Imaging*, Special Issue on Video Object Processing, vol. 11, no. 3, pp. 172–185, Jun. 2005.

[53] Y. Wang, P. M. Jodoin, F. Porikli, J. Konrad, Y. Benezeth, and P. Ishwar, “CDnet 2014: An Expanded Change Detection Benchmark Dataset”, in *2014 IEEE Conference on Computer Vision and Pattern Recognition Workshops*, Jun. 2014, pp. 393–400.

[54] J. Canny, “A Computational Approach to Edge Detection”, *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 8, no. 6, pp. 679–698, Jun. 1986.

[55] C. Harris and M. Stephens, “A combined corner and edge detector”, in *In Proc. of Fourth Alvey Vision Conference*, 1988, pp. 147–151.

[56] H. P. Morevec, “Towards Automatic Visual Obstacle Avoidance”, in *Proceedings of the 5th International Joint Conference on Artificial Intelligence - Volume 2*, ser. IJCAI’77, San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1977, pp. 584–584.

[57] E. Rosten and T. Drummond, “Fusing points and lines for high performance tracking”, in *Tenth IEEE International Conference on Computer Vision (ICCV’05) Volume 1*, vol. 2, Oct. 2005, 1508–1515 Vol. 2.

[58] E. Rosten and T. Drummond, “Machine Learning for High-Speed Corner Detection”, in *Computer Vision – ECCV 2006*, ser. Lecture Notes in Computer Science, Springer, Berlin, Heidelberg, May 2006, pp. 430–443.

[59] D. G. Lowe, “Distinctive Image Features from Scale-Invariant Keypoints”, *International Journal of Computer Vision*, vol. 60, no. 2, pp. 91–110, Nov. 2004.

[60] D. G. Lowe, “Object recognition from local scale-invariant features”, in *Proceedings of the Seventh IEEE International Conference on Computer Vision*, vol. 2, 1999, 1150–1157 vol.2.

[61] H. Bay, T. Tuytelaars, and L. V. Gool, “SURF: Speeded Up Robust Features”, in *Computer Vision – ECCV 2006*, ser. Lecture Notes in Computer Science, Springer, Berlin, Heidelberg, May 2006, pp. 404–417.

[62] M. Calonder, V. Lepetit, M. Ozuysal, T. Trzcinski, C. Strecha, and P. Fua, “BRIEF: Computing a Local Binary Descriptor Very Fast”, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 34, no. 7, pp. 1281–1298, Jul. 2012.

[63] S. Leutenegger, M. Chli, and R. Y. Siegwart, “BRISK: Binary Robust invariant scalable keypoints”, in *2011 International Conference on Computer Vision*, Nov. 2011, pp. 2548–2555.

[64] A. Alahi, R. Ortiz, and P. Vandergheynst, “FREAK: Fast Retina Keypoint”, in *2012 IEEE Conference on Computer Vision and Pattern Recognition*, Jun. 2012, pp. 510–517.

[65] M. Fularz, M. Kraft, A. Schmidt, and A. Kasiński, “A High-Performance FPGA-Based Image Feature Detector and Matcher Based on the FAST and BRIEF Algorithms”, *International Journal of Advanced Robotic Systems*, vol. 12, no. 10, p. 141, Oct. 2015.

[66] E. D. Bello and P. A. Salvadeo, “An image descriptors extraction hardware-architecture inspired on Human Retina”, in *2014 IX Southern Conference on Programmable Logic (SPL)*, Nov. 2014, pp. 1–6.

[67] J. Weberruss, L. Kleeman, D. Boland, and T. Drummond, “FPGA acceleration of multilevel ORB feature extraction for computer vision”, in *2017 27th International Conference on Field Programmable Logic and Applications (FPL)*, Sep. 2017, pp. 1–8.

[68] J. Zhao, X. Huang, and Y. Massoud, “An efficient real-time FPGA implementation for object detection”, in *2014 IEEE 12th International New Circuits and Systems Conference (NEWCAS)*, Jun. 2014, pp. 313–316.

[69] O. Ulusel, C. Picardo, C. B. Harris, S. Reda, and R. I. Bahar, “Hardware acceleration of feature detection and description algorithms on low-power embedded platforms”, in *2016 26th International Conference on Field Programmable Logic and Applications (FPL)*, Aug. 2016, pp. 1–9.

[70] W. Zhu, L. Liu, G. Jiang, S. Yin, and S. Wei, “A 135-frames/s 1080p 87.5-mW Binary-Descriptor-Based Image Feature Extraction Accelerator”, *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 26, no. 8, pp. 1532–1543, Aug. 2016.

[71] P. Viola and M. Jones, “Rapid object detection using a boosted cascade of simple features”, in *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001*, vol. 1, 2001, I–511–I–518 vol.1.

[72] L. A. Hart, *How the brain works: a new understanding of human learning, emotion, and thinking*. New York: Basic Books, 1975.

[73] J.-P. Fillard, *Brain vs Computer: The Challenge of the Century*, 1st. River Edge, NJ, USA: World Scientific Publishing Co., Inc., 2017.

[74] L. O. Chua and L. Yang, “Cellular neural networks: theory”, *IEEE Transactions on Circuits and Systems*, vol. 35, no. 10, pp. 1257–1272, Oct. 1988.

[75] ——, “Cellular neural networks: applications”, *IEEE Transactions on Circuits and Systems*, vol. 35, no. 10, pp. 1273–1290, Oct. 1988.

[76] T. Roska and L. O. Chua, “The CNN universal machine: an analogic array computer”, *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing*, vol. 40, no. 3, pp. 163–173, Mar. 1993.

[77] G. Linan, R. Dominguez-Castro, S. Espejo, and A. Rodriguez-Vazquez, “ACE16k: An advanced focal-plane analog programmable array processor”, in *Proceedings of the 27th European Solid-State Circuits Conference*, Sep. 2001, pp. 201–204.

[78] Z. Nagy and P. Szolgay, “Configurable multilayer CNN-UM emulator on FPGA”, *IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications*, vol. 50, no. 6, pp. 774–778, Jun. 2003.

[79] H. R. Wilson and J. D. Cowan, “Excitatory and Inhibitory Interactions in Localized Populations of Model Neurons”, *Biophysical Journal*, vol. 12, no. 1, pp. 1–24, Jan. 1972.

[80] T. Ayhan and M. E. Yalçın, “Randomly reconfigurable Cellular Neural Network”, in *2011 20th European Conference on Circuit Theory and Design (ECCTD)*, Aug. 2011, pp. 604–607.

[81] T. Ayhan, R. Yeniçeri, S. Ergünay, and M. E. Yalçın, “Hybrid processor population for odor processing”, in *2012 IEEE International Symposium on Circuits and Systems*, May 2012, pp. 177–180.

[82] T. Ahonen, A. Hadid, and M. Pietikainen, “Face Description with Local Binary Patterns: Application to Face Recognition”, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 28, no. 12, pp. 2037–2041, Dec. 2006.

[83] G. D. Field, J. L. Gauthier, A. Sher, M. Greschner, T. A. Machado, L. H. Jepson, J. Shlens, D. E. Gunning, K. Mathieson, W. Dabrowski, L. Paninski, A. M. Litke, and E. J. Chichilnisky, “Functional connectivity in the retina at the resolution of photoreceptors”, *Nature*, vol. 467, no. 7316, pp. 673–677, Oct. 2010.

[84] K.-y. Lee and K.-j. Byun, “A hardware design of optimized ORB algorithm with reduced hardware cost”, Dec. 2013, pp. 58–62.

[85] M. Levoy and P. Hanrahan, “Light Field Rendering”, in *Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques*, ser. SIGGRAPH ’96, New York, NY, USA: ACM, 1996, pp. 31–42.

[86] J. S. Tyo, D. L. Goldstein, D. B. Chenault, and J. A. Shaw, “Review of passive imaging polarimetry for remote sensing applications”, *Applied optics*, vol. 45, no. 22, pp. 5453–5469, 2006.

[87] M. I. Mishchenko, Y. S. Yatskiv, V. K. Rosenbush, and G. Videen, *Polarimetric detection, characterization and remote sensing*. Springer, 2011.

[88] Y. Y. Schechner, S. G. Narasimhan, and S. K. Nayar, “Instant dehazing of images using polarization”, in *Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on*, IEEE, vol. 1, 2001, pp. I–I.

[89] J. Zeil, W. Ribi, A. Narendra, and G. Horváth, “Polarized light and polarization vision in animal sciences”, *ch. Polarisation Vision in Ants, Bees and Wasps*, pp. 41–60, 2014.

[90] N.-w. Cao, W.-q. Liu, and Y.-j. Zhang, “Quantitative study of improvements of the imaging contrast and imaging range by the polarization technique”, 2000.

[91] L. M. Novak, M. C. Burl, W. Irving, and G. Owirka, “Optimal polarimetric processing for enhanced target detection”, in *Telesystems Conference, 1991. Proceedings. Vol. 1., NTC’91., National*, IEEE, 1991, pp. 69–75.

[92] *Polarcam: snapshot micropolarizer cameras user manual*, 4D Technology, 2015, 21 pp.

[93] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “Imagenet classification with deep convolutional neural networks”, in *Advances in neural information processing systems*, 2012, pp. 1097–1105.

[94] K. Simonyan and A. Zisserman, “Very deep convolutional networks for large-scale image recognition”, *arXiv preprint arXiv:1409.1556*, 2014.

[95] Y. LeCun, Y. Bengio, and G. Hinton, “Deep learning”, *nature*, vol. 521, no. 7553, p. 436, 2015.

[96] S. Ren, K. He, R. Girshick, and J. Sun, “Faster r-cnn: towards real-time object detection with region proposal networks”, in *Advances in neural information processing systems*, 2015, pp. 91–99.

[97] J. Dai, Y. Li, K. He, and J. Sun, “R-fcn: object detection via region-based fully convolutional networks”, in *Advances in neural information processing systems*, 2016, pp. 379–387.

[98] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, “Ssd: single shot multibox detector”, in *European conference on computer vision*, Springer, 2016, pp. 21–37.

[99] Tzutalin, *Label image: graphical image annotation tool*, https://github.com/tzutalin/labelImg, Last accessed on 2018-06-01, 2018.

[100] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, *et al.*, “Tensorflow: a system for large-scale machine learning.”, in *OSDI*, vol. 16, 2016, pp. 265–283.

[101] B. Custers, Ed., *The Future of Drone Use: Opportunities and Threats from Ethical and Legal Perspectives*, ser. Information Technology and Law Series. T.M.C. Asser Press, 2016.

[102] İ. Güvenç, O. Ozdemir, Y. Yapici, H. Mehrpouyan, and D. Matolak, "Detection, localization, and tracking of unauthorized UAS and Jammers", in *2017 IEEE/AIAA 36th Digital Avionics Systems Conference (DASC)*, Sep. 2017, pp. 1–10.

[103] B. Harvey and S. O'Young, "Acoustic Detection of a Fixed-Wing UAV", *Drones*, vol. 2, no. 1, p. 4, Jan. 2018.

[104] J. Busset, F. Perrodin, P. Wellig, B. Ott, K. Heutschi, T. Rühl, and T. Nussbaumer, "Detection and tracking of drones using advanced acoustic cameras", in *Unmanned/Unattended Sensors and Sensor Networks XI; and Advanced Free-Space Optical Communication Techniques and Applications*, vol. 9647, International Society for Optics and Photonics, Oct. 2015, 96470F.

[105] F. Gökçe, G. Üçoluk, E. Şahin, and S. Kalkan, "Vision-Based Detection and Distance Estimation of Micro Unmanned Aerial Vehicles", *Sensors (Basel, Switzerland)*, vol. 15, no. 9, pp. 23 805–23 846, Sep. 2015.

[106] K. R. Sapkota, S. Roelofsen, A. Rozantsev, V. Lepetit, D. Gillet, P. Fua, and A. Martinoli, "Vision-based Unmanned Aerial Vehicle detection and tracking for sense and avoid systems", in *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Oct. 2016, pp. 1556–1561.

[107] A. Rozantsev, "Vision-based detection of aircrafts and UAVs", 2017.

[108] P. A. Prates, R. Mendonça, A. Lourenço, F. Marques, J. P. Matos-Carvalho, and J. Barata, "Vision-based UAV detection and tracking using motion signatures", in *2018 IEEE Industrial Cyber-Physical Systems (ICPS)*, May 2018, pp. 482–487.

[109] M. Laurenzis, S. Hengy, A. Hommes, F. Kloeppel, A. Shoykhetbrod, T. Geibig, W. Johannes, P. Naz, and F. Christnacher, "Multi-sensor field trials for detection and tracking of multiple small unmanned aerial vehicles flying at low altitude", in *Signal Processing, Sensor/Information Fusion, and Target Recognition XXVI*, vol. 10200, International Society for Optics and Photonics, May 2017, 102001A.

[110] *Drone detector® | drone detection and protection*, http://dronedetector.com, [Online; accessed 10-September-2018], 2018.

[111] B. Torvik, K. E. Olsen, and H. Griffiths, "Classification of Birds and UAVs Based on Radar Polarimetry", *IEEE Geoscience and Remote Sensing Letters*, vol. 13, no. 9, pp. 1305–1309, Sep. 2016.

[112] M. Ritchie, F. Fioranelli, H. Griffiths, and B. Torvik, "Monostatic and bistatic radar measurements of birds and micro-drone", in *2016 IEEE Radar Conference (RadarConf)*, May 2016, pp. 1–5.

[113] C. Aker and S. Kalkan, "Using deep networks for drone detection", in *2017 14th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS)*, Aug. 2017, pp. 1–6.

[114] V. Popovic, B. Ott, P. Wellig, and Y. Leblebici, “Near-infrared high-resolution real-time omnidirectional imaging platform for drone detection”, in *Target and Background Signatures II*, vol. 9997, International Society for Optics and Photonics, Oct. 2016, p. 999 706.

[115] M. Jacobsen, D. Richmond, M. Hogains, and R. Kastner, “RIFFA 2.1: a Reusable Integration Framework for FPGA Accelerators”, *ACM Trans. Reconfigurable Technol. Syst.*, vol. 8, no. 4, 22:1–22:23, Sep. 2015.

[116] Michael Boxwell, *Solar electricity handbook 2017 edition*, http://www.solarelectricityhandbook.com/solar-irradiance.html, [Online; accessed 10-September-2018], 2017.

[117] *Leon3 processor*, https://www.gaisler.com/index.php/products/processors/leon3, [Online; accessed 22-August-2018].

[118] T. A. Akkaya A. Celik F. and L. Y., “A 10b sar adc with widely scalable sampling rate and agc amplifier front-end”, in *IEEE Nordic Circuits and Systems Conference (NorCAS 2018)*, 2018.

List of publications

- **Ergünay S.** and Leblebici Y., “An Energy-Quality Scalable Smart Camera Architecture for Wireless Vision Sensor Networks”, IEEE Transactions on Circuits and Systems for Video Technology, 2018, (submitted).
- Kilic M., **Ergünay S.**, Leblebici Y., "A Row-Column Accessed Dynamic Element Matching DAC Architecture for SAR ADCs" IEEE Nordic Circuits and Systems Conference (NorCAS 2018), Talinn, Estonia, October 30-31 2018.
- Köse S., **Ergünay S.**, Wellig P., Ott B., Leblebici Y., “Machine Learning Applications in Polarimetric Imaging” SPIE Security + Defense 2018, Berlin, Germany, September 10-13, 2018.
- **Ergünay S.** and Leblebici Y., “A Bio-Inspired Reconfigurable Hardware Implementation for Local Binary Description Applications” The 16th International Workshop on Cellular Nanoscale Networks and their Applications (CNNA 2018).
- **Ergünay S.** and Leblebici Y., “Hardware Implementation of a Smart Camera with Keypoint Detection and Description”, 2018 IEEE International Symposium on Circuits and Systems (ISCAS 2018), pp. 177-180, Florence, Italy, May 20-23, 2012.
- **Ergünay S.** and Leblebici Y., “A Bio-inspired Reconfigurable Architecture for Local Binary Descriptors”, 2016 12th Conference on Ph.D. Research in Microelectronics and Electronics (PRIME), Lisbon, 2016.
- **Ergünay S.**, Popovic V., Seyid K., and Leblebici Y., “A Novel Hybrid Architecture for Real-Time Omnidirectional Image Reconstruction”, In Proceedings of the 9th International Conference on Distributed Smart Cameras (ICDSC ’15), ACM, New York, NY, USA, 152-157.
- Ayhan T., Yeniçeri R., **Ergünay S.**, Yalçın M.E., “Hybrid Processor Population for Odor Processing”, 2012 IEEE International Symposium on Circuits and Systems (ISCAS 2012), pp. 177-180, Seoul, Korea, May 20-23, 2012.
- **Ergünay S.**, Yeniçeri R., Yalçın M.E., “Hardware-Software Co-design of Nonlinear Active Wave Generator with Microblaze Soft Core Processor”, Proc. of 2010 International Sym-

posium on Nonlinear Theory and its Applications (NOLTA2010), pp. 157-160, Krakow, Poland, Sept. 5–8, 2010.

- **Ergünay S.**, Yeniçeri R., Yalçın M. E., "Hardware-Software Co-design of Relaxation Oscillator with Cellular Neural Network on FPGA", Symposium of Innovations in Intelligent Systems and Applications (ASYU 2010), pp. 192-195, Kayseri, Turkey, June 21-24, 2010. (in Turkish)

Selman Ergünay

Contact Information

e-mail: selmanerg@gmail.com
LinkedIn: linkedin.com/in/ergunay

Research Interests

Digital Hardware Design, FPGA, Hardware-Software Co-Design, Embedded Systems
Digital Signal Processing, Image Processing, Multiple Camera Systems

Education

EPFL, École Polytechnique Fédérale de Lausanne, Switzerland

PhD **April 2014 – October 2018**

- *Advisor*: Prof. Yusuf Leblebici
- *Thesis title*: "Smart Camera Architectures for Wireless and Multi-Sensor Vision Applications"

ITU, Istanbul Technical University, Turkey

MSc, GPA: 3.68/4.00 **September 2010 – June 2013**

- *Advisor*: Prof. Müştak Erhan Yalçın
- *Thesis title*: "A Real-Time Bio-Inspired System Based on Olfaction and Stereo Vision"

BSc, GPA: 3.40/4.00 **September 2005 – June 2010**

- *Advisor*: Prof. Müştak Erhan Yalçın
- *Thesis title*: "Hardware-Software Co-Design of Cellular Neural Network on FPGA"

Experience

LSM, Microelectronic Systems Laboratory, EPFL

Research assistant **April 2014 – October 2018**

Teaching assistantship of the following courses: Hardware System Modeling, Lab in EDA Based Design, Digital System Design, Test of VLSI Systems

Intern **September 2013 – April 2014**

Panoptic camera project: Virtex-5 FPGA and QDR-II SRAM based system design for data intensive panoptic camera applications. System level and PCB-schematic design, board level simulation.

TUBITAK, The Scientific and Technological Research Council of Turkey

Researcher at RADAR Division **March 2011 – September 2013**

Implementation of digital signal processing algorithms with FPGA in various RADAR projects.

GSTL, Embedded System Design Laboratory, Istanbul Technical University

Researcher **July 2010 – March 2011**

Microblaze/FPGA based digital system design for Cellular Neural Network applications.

Scholarship

TUBITAK Graduate Scholarship, 2010-2012

Husnu Ozyegin Foundation Success Scholarship, 2005-2010

Technical Tools

System level design:	Python, OpenCV, Matlab, Simulink
Digital hardware design:	VHDL — Xilinx (Vivado, ISE, EDK), Modelsim, GHDL Synopsis Design Vision
Software/Embedded design:	C, C++ — GNU toolchain
PCB design:	Altium
Others:	Linux, Git, Doxygen, LaTeX, Vim