

Universidad de Concepcion
Faculty of Engineering
Department of Electrical Engineering

Advisor:
Dr. Miguel Ernesto Figueroa Toro
Co-advisor:
Dr. Payman Zarkesh-Ha

INTELLIGENT IMAGE SENSOR FOR HETEROGENEOUS SMART CAMERA ARCHITECTURES



Wladimir Elías Valenzuela Fuentealba

Thesis report for the degree of
“Doctor of Science in Electrical Engineering”

Concepción, Chile.
July 2022

Universidad de Concepción
Facultad de Ingeniería
Departamento de Ingeniería Eléctrica

Tutor:
Dr. Miguel Ernesto Figueroa Toro
Co-tutor:
Dr. Payman Zarkesh-Ha

INTELLIGENT IMAGE SENSOR FOR HETEROGENEOUS SMART CAMERA ARCHITECTURES



Wladimir Elías Valenzuela Fuentealba

Informe de tesis para el grado de
“Doctor en Ciencias de la Ingeniería c/m en Ingeniería Eléctrica”

Concepción, Chile.
Julio de 2022

Abstract

In recent years, real-time image processing has gained an essential space in mobile devices, such as IoT endpoints, smartphones, and laptops. This interest is mainly fueled by the need to improve biometrics, such as recognizing different features from humans, security, such as detecting intruders in restricted areas, and safety, such as detecting pedestrians in assisted driving, among others. Mobile devices typically have two essential characteristics, a small form factor, to allow portability, and low energy consumption, to extend battery life. Designing and creating solutions to achieve real-time processing with those two characteristics is not straightforward; moreover if we consider that most image processing methods are not thought to be mobile-friendly. As a consequence, several researchers in the field of hardware design have focused on designing specific purpose circuits. Among these, we can find external digital coprocessors, next to the imager, and smart image sensors (SIS), that add extra circuitry to the imager (i.e., at pixel level) to capture and process the image in the same die.

This thesis report presents the architecture of two smart imaging sensor for face recognition and motion-based object detection, two commonly used image-processing methods. The SISs are based on custom smart pixels capable of computing part of computer vision algorithms in the analog domain, and a respective digital coprocessor that performs the rest of the algorithm in the same die. On the one hand, the SIS for face recognition can compute local spatial gradients in the analog domain, on the smart pixel, and perform image classification on the digital coprocessor. The SIS uses spatial gradients to compute a lightweight version of local-binary patterns (LBP), which was named Ringed LBP (RLBP). The face-recognition method, which is based on Ahonen's algorithm, operates in three stages: (1) it extracts local image features using RLBP, (2) it computes a feature vector using RLBP histograms, and (3) it projects the vector onto a subspace that maximizes class separation and classifies the image using a nearest-neighbor criterion. On the other hand, the SIS for motion-based object detection can compute frame differences in the analog domain, on the smart pixel, and perform morphological operations and connected components to determine the bounding boxes of the detected objects on the digital coprocessor. The smart-pixel array implements on-pixel temporal difference computation using analog memories to detect motion between consecutive frames. The object detection SIS can operate in two modes: (1) as a conventional image sensor and (2) as a smart sensor which delivers a binary image that highlights the pixels in which movement is detected between consecutive frames and the object bounding boxes. The smart pixels were designed the smart

pixel using a $0.18\ \mu\text{m}$ and $0.35\ \mu\text{m}$ mixed-signal CMOS processes. The evaluation of the performance were performed using post-layout parasitic extraction. With a pixel-pitch of $32\ \mu\text{m} \times 32\ \mu\text{m}$, and considering the $0.18\ \mu\text{m}$ and $0.35\ \mu\text{m}$ processes, the fill factor of the face recognition smart pixel is 34% and 76%, respectively, and of 28% and 74% for the object detection smart pixel, respectively. The pixel array for face recognition operates at up to 556 frames per second. Implemented, validated and tested on a Xilinx XC7Z020 field-programmable gate array, the digital coprocessor achieves 96.5% classification accuracy on a database of infrared face images, can classify a 150×80 -pixel image in $94\ \mu\text{s}$, and consumes 71 mW of power. On an array of 320×240 smart pixels, the object detection SIS operates at 60 frames per second. The digital coprocessor was implemented and validated on a Xilinx Artix-7 XC7A35T field-programmable gate array that can run at 125 MHz, detects objects in a frame in $0.614\ \mu\text{s}$, and has a power consumption of 58 mW.

Thanks to the results obtained in this thesis, we can enumerate the following contributions. First, it is possible to design heterogeneous smart cameras that combine smart pixels based on intelligent readout circuits and a digital coprocessor in the same die. Using an array of smart pixels can contribute to exploiting the parallelism present in the image processing algorithms. Designing smart pixels based on intelligent readout circuits allows computing during capture time, thus, reducing processing time and memory resources. Finally, modifying and adapting the image processing algorithms while considering the SIS architecture can lead to simpler methods, mathematically, without significantly impacting results such as precision.

General index

Abstract	i
Figure Index	iv
1 Introduction	1
1.1 General introduction	1
1.2 Hypothesis	4
1.3 General goal	4
1.4 Specific goals	4
1.5 Scope	5
1.6 Contributions	5
1.7 Report organization	5
2 Related work	7
2.1 General discussion	9
3 Methods	11
3.1 Face recognition	11
3.1.1 Face recognition background	11
3.1.2 Face recognition for SIS	12
3.2 Object detection	16
3.2.1 Object detection background	16
3.2.2 Object detection for SIS	17
4 SIS architectures	23
4.1 SIS architecture for face recognition	23
4.1.1 Smart pixel	23
4.1.2 RLBP generator	28
4.1.3 Digital coprocessor	28
4.2 SIS architecture for motion-based object detection	32
4.2.1 Smart pixel	33
4.2.2 A-THR	37
4.2.3 Digital coprocessor	38



5	Results	41
5.1	Evaluation Methodology	41
5.2	Face recognition results	42
5.2.1	Smart pixel and RPG implementation	43
5.2.2	FPGA implementation of the digital coprocessor	46
5.2.3	Method classification performance	47
5.2.4	SIS classification performance	50
5.3	Object detection results	51
5.3.1	Smart pixel and A-THR implementation	51
5.3.2	FPGA implementation of the digital coprocessor	53
5.3.3	SIS object detection performance	55
5.3.4	Comparison to Related work	57
6	Conclusions	61
6.1	Future work	63



Figure Index

3.1	Proposed method and SIS architecture. The left hand side illustrates the steps of the presented face recognition algorithm. The right hand side shows the elements of the proposed SIS, namely smart-pixel array, pattern generator and digital coprocessor, which execute the stages of the algorithm.	13
3.2	Examples of LBP and RLBP operators on a 3×3 -pixel window.	14
3.3	Ahonen's algorithm using uniform RLBP.	14
3.4	Illustration of the motion-based object detection algorithm and SIS architecture.	17
3.5	Results of the steps of motion-based object detection: (a) input image, (b) result of the frame-difference, (c) thresholding the frame-difference, (d) applying the image-open morphological transformation, and (e) bounding boxes.	19
3.6	Graphic description of (a) conventional image erosion and (b) binary image erosion. Conventional image erosion replaces central pixel with the lower value on its neighborhood. Binary image erosion replaces central pixel with a logical 0 if there is at least one logical 0 on its neighborhood.	20
3.7	Graphic description of (a) conventional image dilation and (b) binary image dilation. Conventional image dilation replaces central pixel with the higher value on its neighborhood. Binary image dilation replaces central pixel with a logical 1 if there is at least one logical 1 on its neighborhood.	20

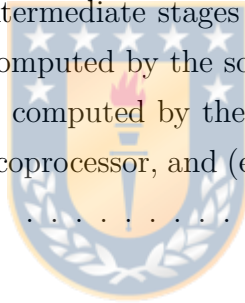
- 3.8 Graphic description of the connected components algorithm. The algorithm receives as input a binary image with movement pixels from the frame-difference stage. The algorithm defines a 2×2 -pixel window, corresponding to the analyzed pixel, and its west, northwest and north neighbors. For each pixel, if its three neighbors are not labeled as part of an object, then the algorithm assigns a new label to the pixel (Figs. a and b). If one or more of the neighbors have the same label, the algorithm assigns the same label to the pixel (Fig. c). When two or more neighbors have different labels, the algorithm assigns one of the labels to the pixel and update the equivalence table to reflect that all those labels now belong to the same connected component (Fig. d). 21
- 4.1 Architecture of the proposed SIS. An array of smart pixels outputs either the pixel value or the difference between horizontally adjacent pixels. An Ringed Local Binary Pattern (RLBP) generator (RPG) reads pixel values and creates an 8-bit RLBP for each pixel in the image. . The digital coprocessor computes histograms of RLBP patterns to construct the feature vector, executes the LDA projection on each vector, and selects the nearest neighbor from a stored set of projected vectors using the Euclidean distance. 24
- 4.2 The smart pixel consists of an analog input-select multiplexer, a configurable Capacitive Transimpedance Amplifier (CTIA), and a row-select switch. All the smart pixels in the array share the control signals placed above in the figure, and all the pixels in the same column share the *Column output* signal. The input to the CTIA can be selected from the photodetector in the local or adjacent pixel. 25
- 4.3 Schematic diagram of the configurable CTIA. The CTIA integrates the photodetector currents and outputs a voltage that represents either the pixel value or the difference between horizontally-adjacent pixels. 25
- 4.4 Smart pixel in conventional mode: the input-select switches pass the current from PD1, *sw1* and *sw4* are closed to integrate the current, and *sw2* and *sw3* stay open. 26
- 4.5 Simplified view of positive and negative integration. During positive integration: *sw1* and *sw4* stay closed, *sw2* and *sw3* stay open. During positive integration: *sw2* and *sw3* stay closed, *sw1* and *sw4* stay open. 26

- 4.6 Architecture of the RPG. An input comparator compares the local gradient value for each pixel to a reference voltage. The digital comparator outputs are sequentially stored in an array of 3×3 flip-flops, organized as 3 shift registers. The RPG outputs an 8-bit RLBP with the output of all the flip-flops except for the one at the center. 27
- 4.7 Architecture of the digital coprocessor. The processor receives a stream of RLBP's from the RPG, and simultaneously builds the histogram vector and projects it using LDA. A memory controller retrieves the LDA coefficients from RAM. The Euclidean distance between the projected vector and the contents of database of stored faces is used for classification with the nearest-neighbor criterion. 29
- 4.8 Architecture of the LDA projection module. The module transforms the 8-bit RLBP into a 6-bit uniform RLBP (uRP). For each uRP value received, the module accumulates the value of its corresponding LDA coefficient, thus performing histogram computation and LDA projection in a single step. 30
- 4.9 Euclidean distance module. It normalizes and centers the input vector and computes the distance between vectors p and q as $\sum p_i^2 - 2 \sum p_i q_i + \sum q_i^2$ 31
- 4.10 Classification module. The module implements a nearest-neighbor criterion by selecting the face label that corresponds to the minimum distance computed between the input image and the stored database of know faces. 32
- 4.11 Architecture of the proposed SIS. An array of smart pixels outputs either the pixel value or frame-difference. The A-THR module determines whether the absolute value of the frame-differences exceeds an application-defined threshold. The digital coprocessor computes image opening to improve the object detection, and uses a connected components algorithm to detect objects in the image and compute their bounding boxes. The digital coprocessor can be configured to output the original image or the binary image and the bounding boxes for the objects. 33
- 4.12 Architecture of the smart pixel. $Negtin$, $PostInt$, $BuffSL$ and $Vbias$ are global bias and control signals. $Row\ control$ is shared by all the pixels in a row and $Column\ output$ is shared by all the pixels in the same column. 34

- 4.13 Configurable CTIA. The output voltage of the CTIA represents either the pixel value or the difference between the pixels in the current and past frame. The configurable CTIA includes two integration capacitors C_{int1} and C_{int2} of equal size, which are used as double buffers to integrate and compute the frame difference. 34
- 4.14 Smart pixel in conventional mode: the input-enable switch passes the current from the photodiode PD1, $sw1$, $sw4$ and $sw5$ are closed to integrate the current using C_{int1} , and $sw2$, $sw3$ and $sw6$ stay open. 35
- 4.15 Simplified view of the CTIA in frame-difference mode during odd and even frames. During an odd frame, $sw2$ and $sw3$ are closed while $sw1$ and $sw4$ are open. During the store phase, $sw5$ is open and $sw6$ is closed, and during the subtract phase the states of $sw5$ and $sw6$ are reversed. At the end of the frame, the voltage across C_{int1} represents the frame-difference between the current and previous frame. During even frames the state of all switches is the complement of the odd frames, and the frame-difference is represented by the voltage across C_{int2} 36
- 4.16 Architecture of the A-THR. An input comparator compares the frame-difference for each pixel to two reference voltages. The comparator OA1 outputs a logical 1 when $V_{pixel} > V_{th}^+$, and the comparator OA2 outputs a logical 1 when $V_{pixel} < V_{th}^-$. A logical OR outputs a logical 1 if one of the two conditions is met. 38
- 4.17 Architecture of the digital coprocessor. The coprocessor receives a stream of movement pixels, applies morphological opening operation (erosion+dilation), and computes the connected components of the resulting binary image and their bounding boxes. 39
- 4.18 Image erosion. The module uses two line buffers and six registers to define a 3×3 -pixel window from the output of the smart pixel array, and performs image erosion by computing a logical AND operation between them. 39
- 4.19 Image dilation. The module uses two line buffers and six registers to define a 3×3 -pixel window from the output of the image erosion module, and performs image dilation by computing a logical OR operation between the pixels. 39

4.20	Architecture of the connected components module. First, it analyzes the current pixel and its north, northwest and west neighbors, determining which movement pixels are connected. The module assigns a label to the current pixel and maintains an equivalence table to merge connected components in the image. The module also computes the bounding boxes for all connected components and merges them using the equivalence table.	40
5.1	Experimental setup to test the face-recognition algorithm. An FPGA board receives IR images from a FLIR Tau 2 camera core and uses a HOG algorithm to detect face locations. The FPGA emulates the smart pixel array and the digital coprocessor. A monitor connected to the FPGA displays the image acquired by the smart pixel array, and the location and number of identified faces. The FPGA sends the labels of the recognized faces to a remote computer via Ethernet. . . .	42
5.2	Layout of the smart-pixel. This work used the design shown in Fig. 4.3, implemented on the TMS320C64x 0.35 μm mixed-signal process. The opamp and integration capacitors are implemented using two poly layers.	43
5.3	Post-layout simulation of five pixels in the SIS operating in local-gradient mode. The graph shows the voltage across the integration capacitor of the CTIA in Fig. 4.14 during the positive and negative integration phases shown in Fig. 4.5. . .	45
5.4	Post-layout simulation of the RPG input comparator while reading multiple pixels. The plot shows the integration phase for the first and third pixel, the voltage input to the comparator, the reference voltage, and the voltage output. Gradient values are read every 50ns.	46
5.5	Effect of V_{ref} in the RLBP values generated by the RPG for 3 images acquired using a FLIR Tau 2 thermal IR core. (a) original IR image, (b) RLBP image generated by software, (c)-(e) RLBP images generated by the RPG for V_{ref} values of 1.665 V, 1.650 V and 1.645 V, respectively.	50
5.6	Classification accuracy as a function of the value of the comparator input V_{ref} in Fig. 4.6.	51
5.7	Diagram of the smart-pixel layout. Using the design shown in Fig. 4.3 and implemented on the TMS320C64x 0.35 μm mixed-signal process. The opamp and integration capacitors are implemented using two poly layers.	52

- 5.8 Post-layout simulation of a pixel in the SIS operating in frame-difference mode. The graph shows the voltage across the two integration capacitors of the CTIA during two consecutive frames. 53
- 5.9 Post-layout simulation of the A-THR comparator while reading multiple pixels in frame-difference mode. The plot shows the subtract phase for two pixels A and B in the same column, during an odd frame. In the readout phase, the comparator consecutively samples all pixels in each column, comparing their value to the application-defined thresholds and output a logic 1 when the movement in the pixel exceeds the threshold. Pixel values are sampled every 50 ns. 54
- 5.10 Visual comparison of the intermediate stages of the algorithm on the software and analog section of the SIS: (a) input frame, (b) frame-difference computed by the software, (c) software output after thresholding, (d) smart-pixel array output in frame-difference mode, and (e) A-THR output in the SIS. 57
- 5.11 Visual comparison of the intermediate stages of the software and digital coprocessor: (a) image opening computed by the software, (b) bounding boxes in the software, (c) image opening computed by the digital coprocessor, (d) bounding boxes output by the digital coprocessor, and (e) comparison between the outputs of the two implementations. 58



Acronyms

SIS Smart Image Sensor

IoT Internet-of-Things

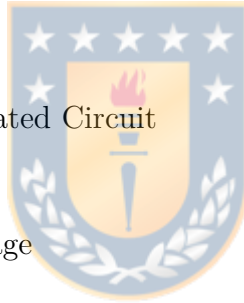
CIS CMOS Image Sensor

CTIA Capacitive Transimpedance Amplifier

SVM Support Vector Machine

iROIC Intelligent Read Out Integrated Circuit

HDL Hardware Description Language



RTL Register-Transfer Level

FDI Fully-digital implementation

LUT Lookup table

FF Flip-Flop

BRAM Block Random Access Memory

IR Infrared

LBP Local Binary Pattern

RLBP Ringed Local Binary Pattern

ASIC Application-Specific Integrated Circuit

FPGA Field Programmable Gate Array

FPA Focal Plane Array

SVM Support Vector Machine



1. Introduction

1.1 General introduction

Computer vision is a discipline that has gained an important place in data analysis on various scientific and industrial applications. Among the applications of computer vision are obstacle detection [1] and position and speed estimation for accident avoidance [2] in driverless cars, pedestrian detection using infrared cameras for surveillance [3,4], autonomous underwater monitoring system for detecting life on the seabed [5], improvement of food industry using real time smart machines and predictable models [6], real-time pupil localization for drive safety improvements [7,8], among others.

The scientific and industrial community have recent interest in image-based biometric methods that fostered a growing interest in smart image systems that can handle the computational requirements of real-time video analysis. Biometrics is described as a pattern-recognition technique for individual identification, based on their physical, chemical or behavioral characteristics [9,10]. One of the most popular biometric techniques is face recognition [10], which has abundant applications [11–13] in various areas, such as: (1) security, including identity verification [14,15], computer or mobile device unlock [15,16], criminal records search, and voter registration; (2) surveillance, such as cameras used on closed circuit television [17]; and (3) access control, that could grant access to a specific place or an electronic account to a group of people [18] using their faces as a credential.

On the other hand, there is a growing interest in image and video processing on mobile devices [19], including a variety of approaches for biometric recognition [20], fueled by the commercial interest in robust authentication methods for smartphones, laptops, tablets and other mobile devices [15,16], and mobile object detection [21–23]. One of the key advantages of mobile devices in general is portability, mainly supported by a small form factor and a low energy consumption to extend the device battery life [24]. Computer vision typically requires a huge amount of computation power to be capable of delivering highly precise and fast results, relying on increased computational requirements for improved accuracy [25,26]. This increment on computational requirements is counterproductive when the objective is to implement computer vision applications into resource-constrained platforms, such as mobile devices or IoT endpoints, as the high computation capabilities affect form factor and power consumption, requiring major

efforts to achieve high-performance mobile solutions [27]. Thus, it is important to take into account different considerations when selecting suitable hardware for computer vision tasks [25].

In particular for low-power biometrics, different sensors have been reported in the literature, such as adaptive wireless body sensor networks for biometrics and healthcare applications [28] for long-time monitoring, sensors for age and gender classification which monitor brain signals using electroencephalography [29], biometric recognition systems for mobile Internet-of-Things (IoT) devices [30], and an ultra-low-power hybrid face recognition processor integrated with a CMOS Image Sensor (CIS) [31] applied to mobile devices [32, 33]. Common to all these designs are two technological challenges: low power consumption and circuit area reduction. Both are intimately related to key features of mobile devices, such as energy autonomy and size [28].

To reduce power consumption and increase hardware integration, designers often turn to dedicated hardware architectures specifically designed to perform a singular task of interest. In image processing, there is visible progress in the development of Smart Image Sensors (SISs), also referred to as vision chips. The SIS are dedicated electronic devices that combine conventional image sensors with additional circuitry on the same die [34]. The additional circuitry performs, either partially or totally, operations and algorithms associated with different image processing methods. It is possible to organize image processing methods and hardware into three levels, depending on where the data is processed: pixel level processing, column/row level processing and data-sequence level processing, i.e., processing that occurs outside the pixel array after digital conversion [34]. Of these three levels, the most challenging when designing the architecture of a SIS, is the pixel level, because there is a very limited area available inside each of the pixels, and the design of the processing circuits must minimize the overhead imposed on the pixel size. Therefore, the complexity of the circuitry is limited by the space that can be occupied and, as a consequence, the complexity of the image-processing operations that can be introduced into the pixel is also limited. This difficult tradeoff can be observed on the SIS work available the literature. Some examples of image processing methods implemented as a SIS are edge detection [35–37], image classification using analog lightweight convolutional neural networks [38], on-chip non-uniformity compensation on IR image sensors [39, 40], target tracking [35, 37], motion detection [35], feature extraction [41–45], and face recognition [31–33], among others [46].

When a pixel performs a significant level of data processing, such as complex mathematical operations or feature extraction, then it is frequently referred to as a smart pixel. Smart pixels have the capacity to deliver a high level of fine-grained parallelism (spatial or temporal), where each smart pixel of the Focal Plane Array (FPA) in a SIS performs computation simultaneously

on different data [34]. Fine-grained parallelism can improve the execution time of the algorithm, lower the processing latency, reduce the amount of memory required to store temporary results, and maximize output data throughput [43,44]. Moreover, when the smart pixel operates in the analog domain, they can also reduce power consumption and die area [47].

The main problem presented in this thesis can be elaborated as follows. Portable devices require to accomplish two characteristics: 1) low power consumption to extend battery life and 2) a small-form factor. As there is a growing interest in real-time image processing for applications such as security and authentication, the logical path is the implementation of these applications in portable devices. As technology continuously improves, an attractive alternative is to pushing its limits to reduce power consumption and form-factor, including implementing real-time image processing algorithms in portable devices. A natural way of achieving low power and portability while processing in real-time is using dedicated hardware, such as FPGAs or ASICs, that exploits the parallelism available in the algorithm. These alternatives are typically connected to the output of the imager. Thus they are still connected in a serial-data stream, considerably reducing the capacity of exploiting the parallelism and then limiting the capacity to further reducing power consumption and form factor. Another alternative are SISs, which are heterogeneous devices composed of an imager and processing circuitry in the same die. This heterogeneity allows computing part of the algorithms between the circuitry at the pixel- or column-level of the sensor array and a coprocessor (analog or digital). Including pixel or column-level processing allows to exploits the parallelism available in the algorithms. However, many solutions require first capturing the image, i.e. waiting for the capture time of the readout circuit, to later process the pixel data. There is an open field that can further exploits the parallelism and power consumption, that is processing the images during the capture time.

This thesis reports the design and evaluation of two novel Intelligent Read Out Integrated Circuits (iROICs) and their complete SIS architecture for two algorithms: face recognition and motion-based object detection. The presented iROICs are designed to compute part of the algorithms during the current-to-voltage integration process of the capture time. The proposed SISs are composed of a smart-pixel architecture based on a Capacitive Transimpedance Amplifier (CTIA) integrator, widely used on thermal IR image sensors. Therefore, the iROIC is suitable for both face recognition and motion-based object detection in thermal IR and visible images. The SIS design is suitable for mobile devices, where the SIS can be operated as a conventional sensor to capture images of a scene, or, depending on the SIS architecture, as a face recognition system to obtain the identity of a subject or as a object detection system based on motion. Using a CTIA-based pixel, it is possible to add a small number of transistors, thereby minimizing the added cost in area and power compared to a regular image sensor.

1.2 Hypothesis

The hypothesis of this thesis is that *a smart-pixel array can efficiently distribute the computation of an image processing system between pixel level, column level and a digital coprocessor, and perform highly-parallel fine-grain calculations for local gradient-based image processing algorithms in the spatial and the time domains.*

An smart-pixel array, next to a configurable digital processor that interfaces with it, can allow to fabricate an SIS capable of implement computer vision algorithms distributed between both domains. The SIS can distribute the calculation of different computer vision algorithms by performing highly-parallel tasks all along the smart pixel array. This distribution can increase the overall performance of the smart image system with a penalty on the ratio between the pixel circuit and photodetector areas (fill factor) that can be comparable to the fill factor conventional readout circuit.

1.3 General goal

The general goal of this thesis is to design, test, evaluate and probe that an SIS, that computes part of computer vision algorithms on-pixel can improve the performance of the algorithm with a small penalty on the pixel fill factor.

1.4 Specific goals

1. Define the digital image processing algorithms on the literature to identify an algorithm that relies on operations at the pixel-level.
2. Design and adapt different active pixel architectures to execute on-pixel operations that are part of image processing algorithms.
3. Generalize a design of a smart pixel interfaced with a configurable digital processor. Design a custom digital architecture processor to interface with the designed smart pixel architecture and compute the desired pixel-level operations.

1.5 Scope

The scope of this thesis work includes: 1) research and propose SISs as circuits and image processing systems, 2) simulate and evaluate the performance (speed, power, utilization) of the proposed SISs, and 3) publish the results as articles on Web of Science indexed journals.

As the main goal is to design an SIS architecture, the algorithms that are being evaluated and reviewed during this thesis are limited to classic and mathematically simple available in the literature. This scope reduces the literature analysis to those works that are related to the two applications implemented, that are face recognition and object detection, and can be thought of as part of a heterogeneous smart camera architecture.

1.6 Contributions

Thanks to the results obtained in this thesis, we can enumerate the following contributions:

1. It is possible to design heterogeneous smart cameras that combine smart pixels based on intelligent readout circuits and a digital coprocessor in the same die
2. Using an array of smart pixels can contribute to exploiting the parallelism present in the image processing algorithms.
3. Designing smart pixels based on intelligent readout circuits allows computing during capture time, thus, reducing processing time and memory resources.
4. Modifying and adapting the image processing algorithms while considering the architecture can lead to simpler methods, mathematically, without significantly impacting results such as precision.

1.7 Report organization

The rest of this thesis report is organized as follows: Chapter 2 discusses related work. Chapter 3 describes the face recognition and object detection methods used in to desing the SISs. Chapter 4 describes the proposed SISs architecture, including their smart pixel, the coprocessor and digital

controller. Chapter 5 presents the algorithm and SISs performance results. Finally, Chapter 6 concludes with a discussion of the results, summarizes key contributions of this thesis and its possible outcomes.



2. Related work

The technological advances in high-performance computing have enabled the development of fast and highly-accurate computer vision systems, including face recognition and object detection. Most frequently, this performance is achieved using power-hungry processors and graphics processing units (GPUs) [48, 49]. While this cost in power and space may not be important in big-data applications that require high precision, it is normally not acceptable in mobile or portable systems [20], which require compact, power-efficient electronics. For these applications, modern technologies have enabled the design of special-purpose processing systems on dedicated hardware that achieve high speed and portability with low power. These designs are implemented on programmable devices such as field-programmable gate arrays (FPGAs) or Application-Specific Integrated Circuit (ASICs). This chapter summarizes related work in face recognition and object detection. The first two sections enumerate and describe FPGA and SIS implementations, and the final section discusses the reviewed literature.

Considering FPGA devices, researchers have developed special systems focused on a wide variety of computer vision methods. FPGAs are popular implementation platforms because of their higher level of fine-grained parallelism and lower power consumption compared to traditional software-programmable solutions.

In the case of facial recognition, many of the FPGA implementations have been developed to focus on speed, portability and low power-consumption. A popular method for facial and object recognition is the use of different type of neural networks, based on Histogram of Oriented Gradients (HOG) [50], a combination of Weighted Modular Principle Component Analysis (WMPCA) and a Radial Basis Function Neural Network (RBFNN) [51] and Convolutional Neural Networks (CNN) [52]. A common disadvantage of these solutions is the limited capacity of on-chip FPGA memory, which is insufficient to store the large number of parameters required by the CNN. Storing these parameters in external memory reduces the throughput of the implementation, therefore limiting the application of FPGAs for small form-factor face recognition in real time. Other algorithms have been addresses that uses local information, such as face recognition algorithms based on Fast Fourier Transform (FFT) [53] or Local Binary Pattern (LBP) and linear discriminant analysis (LDA) [54], which can help to minimize the memory usage and can increase the recognition time.

On the object location and classification side, during recent years many FPGA-based CNN

architectures have been proposed [55–59]. As described above, a common disadvantage of neural networks solutions is the limited capacity of on-chip FPGA memory, and limits the application of FPGAs for small form-factor object detection in real time. This issue was addressed by Long et al. [60], who implemented an FPGA-based object detection algorithm based on multi-frame information fusion. Their algorithm uses a reduced number of parameters for HOG-based object location and Support Vector Machines (SVMs) for classification, and achieves a throughput of up to 10,000 fps. Nakahara et al. [61] presented an object detection algorithm based on a multiscale sliding-window location search, which binarizes the CNN parameters to reduce their memory requirements, and enables the implementation of the complete network using only on-chip memory. Despite the throughput improvement achieved by using on-chip parameters, all the solutions described above read the image pixels as a serial stream from the image sensor. This has the effect of increasing the latency and limiting the data parallelism available to the algorithm, compared to having access to all the pixels simultaneously. Moreover, algorithms that access the image data serially require line buffers or even entire frame buffers, further increasing the memory requirements of the hardware platform.

Despite the throughput improvement achieved by using on-chip parameters, all the FPGA-solutions described above read the image pixels as a serial stream from the image sensor. This read method has the effect of increasing the latency and limiting the data parallelism available to the algorithm, compared to having access to all the pixels simultaneously. Moreover, algorithms that access the image data serially require line buffers or even entire frame buffers, further increasing the memory requirements of the hardware platform.

As discussed in Chapter 1, SISs are special-purpose image sensors that combine conventional imagers with additional circuitry to process pixel data on the same chip. When an SIS is designed with computational circuits in every pixel (smart pixels), it can exploit the pixel-level parallelism available in the image-processing algorithm. This parallelism shortens latency, increases throughput, and reduces the memory requirements of the solution [44,62–64]. To further reduce power and area, SISs typically use analog circuits to store and process the data [62,65]. For example, Lee et al. [65] presented an SIS with embedded object detection that uses a reconfigurable pixel array capable of computing frame differences and spatial gradients. The SIS uses a capacitor in every pixel that acts as an analog memory to compute frame differences. Choi et al [62] use a similar approach to implement motion-triggered object detection. To reduce circuit area and improve fill factor, they use the same capacitor for two horizontally adjacent pixels and alternate its use between odd and even frames, thus trading motion-detection horizontal resolution for fill factor. An alternative technique to improve fill factor is to perform computation during photocurrent integration using an intelligent Readout Circuit (iROIC) [66]. For

example, the SIS presented by Gottardi et al. [67] computes local gradients using this technique to implement a lightweight version of local-binary patterns (LBP). Our own previous work [64] also uses an iROIC to compute face recognition in visible-range and IR image sensors using an array of smart pixels based on a configurable CTIA, reducing precision by only 1% compared to a fully digital implementation.

Implementing most of the computation at the pixel level using smart pixels reduces the die area available for the photodetector in each pixel, thus reducing the fill factor of the imager. To mitigate this effect, several SISs implement part of the computation at the column level, thus improving fill factor at the cost of reducing parallelism and increasing memory requirements. For example, Jin et al. [36] designed an SIS that computes edge detection using column-level circuits and static memory. Young et al. [68] presented an SIS for object detection that combines pixel- and column-level processing to compute image features based on HOG. Their SIS eliminates redundant illumination data during readout, thus compressing the HOG feature descriptors by up to 25 times compared to a conventional 8-bit readout. Kim et al. [33] detect and recognize faces by combining, on a single chip, a standard imager architecture and a mixed-signal CNN that implements its first layer in the analog domain. Computing part of the operations of the algorithm using analog circuits degrades the accuracy by 1.3%, but it also reduces the power consumption by 15.7% because Analog-to-Digital Converters (ADCs) are one of the most power-consuming elements in standard CIS [69]. The SIS presented by Zhong et al. [70] computes edge detection and omnidirectional LBP using column-level circuits and array of capacitors capable of storing two rows of the image. Our own previous work [64] computes LBP using a combination on-pixel and column-level processing. An array of smart pixels performs the comparisons between adjacent pixels and outputs a binary value, which is used by column-level circuits to construct the LBP features. The single-bit output of the smart pixel allows us to reduce the memory required by the line buffers and the time to read the data from the pixel array and improves the fill factor by moving a significant part of the computation to the column-level circuit.

2.1 General discussion

The FPGA solutions described and analyzed above show that memory utilization is a key concern for low power, real-time and small form factor solutions for face recognition and object detection. The analysis shows that SISs are a solid alternative to further achieve the power and performance that is required by computer vision on mobile devices. Despite having drawbacks

on precision and fill factor, SISs are competitive compared to their fully digital counterparts.

Most SISs reviewed in this work shows that local filters and local algorithms, such as median filter or LBP, are a recurrent objective. Some of these address local kernels inside the smart pixel, but their small fill-factor difficults their scalability for high-resolution systems. Furthermore, most of the SISs with competitive fill factor are those that compute major part of the computer vision algorithm in their respective digital coprocessor.

The SIS architectures described above use different techniques to integrate computation into the image sensor efficiently, including column-level processing, computing in the analog domain, limiting LBP kernel size and reducing the number of comparisons in the kernel. These tradeoffs mainly aim to reduce computation time and maximize fill factor. It is also important to note that some of the computation can be performed at integration time, without waiting for the entire image to be acquired. Indeed, Gottardi et al. [71] computes the difference between neighboring pixels during integration to obtain a simplified version of LBP. The work presented by Young et al. [68] indicates that compressing data prior to extracting it from the readout reduces the bandwidth for data transmission and computation. This reduction is reflected directly as overall energy savings. The work presented by Lee et al. [65] demonstrates that a capacitor used as an analog memory is sufficient to store previous frame data to compute the frame-difference step of motion-based object detection.

From the point of view of face recognition, the literature shows that infrared face recognition is a good option for enhanced security or PAD [72]. Popa et al. [73] improve PAD performance using a combination of IR and conventional cameras. Hoon et al [74] proposed NIRFaceNet, a variation of the FaceNet method tailored for NIR images. Tested on different NIR data sets, NIRFaceNet achieves accuracies between 73.1% and 94.8%. Hermosilla et.al [75] tested different methods of face recognition on two thermal IR databases, and they achieved their best accuracies using Gabor Jet Descriptors (96.6%), Weber Local Descriptors (94.9%), and LBPhistograms (92.0%).

The discussion above shows that SISs are a viable alternative to digital processors to achieve the low power and high performance required by computer vision on mobile devices, while achieving comparable precision [33, 64]. An SIS can exploit the pixel-level parallelism of the algorithm, but the area used by the processing circuits limits the fill factor of the SIS. This limitation can be mitigated by performing part of the computation during photocurrent integration and by moving computation to column-level circuits. The design presented in this paper uses both techniques to build a two-mode imager that operates as a conventional sensor and computes object location, using a configurable CTIA suitable for the thermal IR range.

3. Methods

3.1 Face recognition

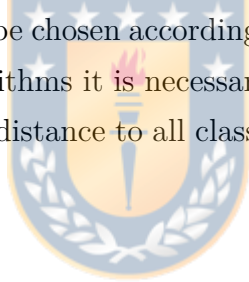
3.1.1 Face recognition background

According to [76], a face recognition system is useful when it can work on both videos and images, processes in real time, and is independent of the person while being robust to different light conditions and different angles. The three fundamental steps of face recognition algorithms are face detection, feature extraction and face classification, but typically the algorithms assume that the face is already detected or posed at the foreground of the image. The feature extraction step uses face images to generate a set of feature vectors that describe it. These vectors represent, depending on the method, different features of the face such as textures, face parts (mouth, nose, etc.), and their geometric distribution. With these, it is possible to have a general representation of the face structure.

Based on the focus of the data processing, feature extraction algorithms can be categorized as local (or geometric) or as holistic. On the one hand, local algorithms consider the general structure of faces, that is, discover distinctive features using appearance-based or key-points-based techniques. LBP is an appearance-based method that has proved to obtain competitive performance compared to the literature on visible face databases thanks to its invariance to the rotation of the target image [77–79]. Other LBP variations can be found that improve the base LBP method such as multiscale LBP [80] or local ternary pattern (LTP) [81]. Histogram of oriented gradients (HOG) is another appearance-based descriptor widely used in literature [82–84], which consist of divide the face image into small regions and generate a histogram of pixel gradients. Then, the histograms of the whole regions are combined to extract the feature of the face image. Other local feature extraction methods can be found such as Speeded Up Robust Features (SURF) [85, 86], Locality Preserving Projections (LPP) [85, 87] or multimodal deep face recognition (MDFR) [88]. Scale-Invariant Feature Transform (SIFT) is a well-known key-points-based technique that extracts a number of meaningful descriptive image features [85, 89] which are invariant to scaling, rotation and illumination. The main idea of SIFT is that the relative position of key points must remain the same in different images of the same face.

On the other hand, holistic algorithms consider that any face-image collection has redundant information that can be discarded by applying the tensor’s decomposition [11]. With this, these algorithms generate a new collection of fundamental vectors that represents the original collection into a subspace of smaller dimension, while preserving the original set of images. There is a wide variety of holistic methods in the literature. Principal component analysis (PCA) or Eigenfaces [90] is a well-known method that efficiently characterizes the facial images in several works [91,92]. They proved that using a standard facial image (eigenpicture) and only a few weights for each facial image in the database it could recreate any of them very closely. Other common method is linear discriminative analysis (LDA) or Fisherfaces [93], which is robust to light variances and facial expressions by projecting the faces into a small-dimensional subspace that generates well-isolated classes. Independent component analysis [94] (ICA) is a third well-known method that consists of a PCA generalization where the most important information can be included in a high-arrange relationship between pixels.

The classification stage uses the feature vector to label the image, using methods such as nearest neighbors [95,96], Support Vector Machines (SVMs) [96,97], or deep neural networks [98, 99]. The classification method must be chosen according to the feature extracted in the previous stage. For example, on holistic algorithms it is necessary to first calculate the projection of the captured face and then calculate its distance to all classes described in that subspace using any distance metric.



3.1.2 Face recognition for SIS

Figure 3.1 depicts a block diagram of the SIS designed in this work. The left hand side of the figure shows the steps of the proposed face classification algorithm, which is described below. The right hand side of the figure relates each step of the algorithm to the component of the proposed SIS that implements it. The architecture of the SIS is described in Section 4.1.

Algorithm 1 describes the proposed face recognition method, which is based on Ahonen’s LBP-based algorithm [100]. The feature extraction stage replaces LBP with the custom RLBP descriptor and projects the feature vector onto a reduced space using LDA. The classification stage compares the projected vector to a stored database of known faces and selects an ID for the input image using a nearest-neighbor criterion. As shown in Algorithm 1, the feature extraction stage first computes an 8-bit RLBP value for each pixel in the image using Algorithm 2. Figure 3.2 compares regular LBP to RLBP for a 3×3 -pixel kernel: LBP, shown in Fig. 3.2a, compares each pixel to its 8 neighbors and concatenates the results to build a binary

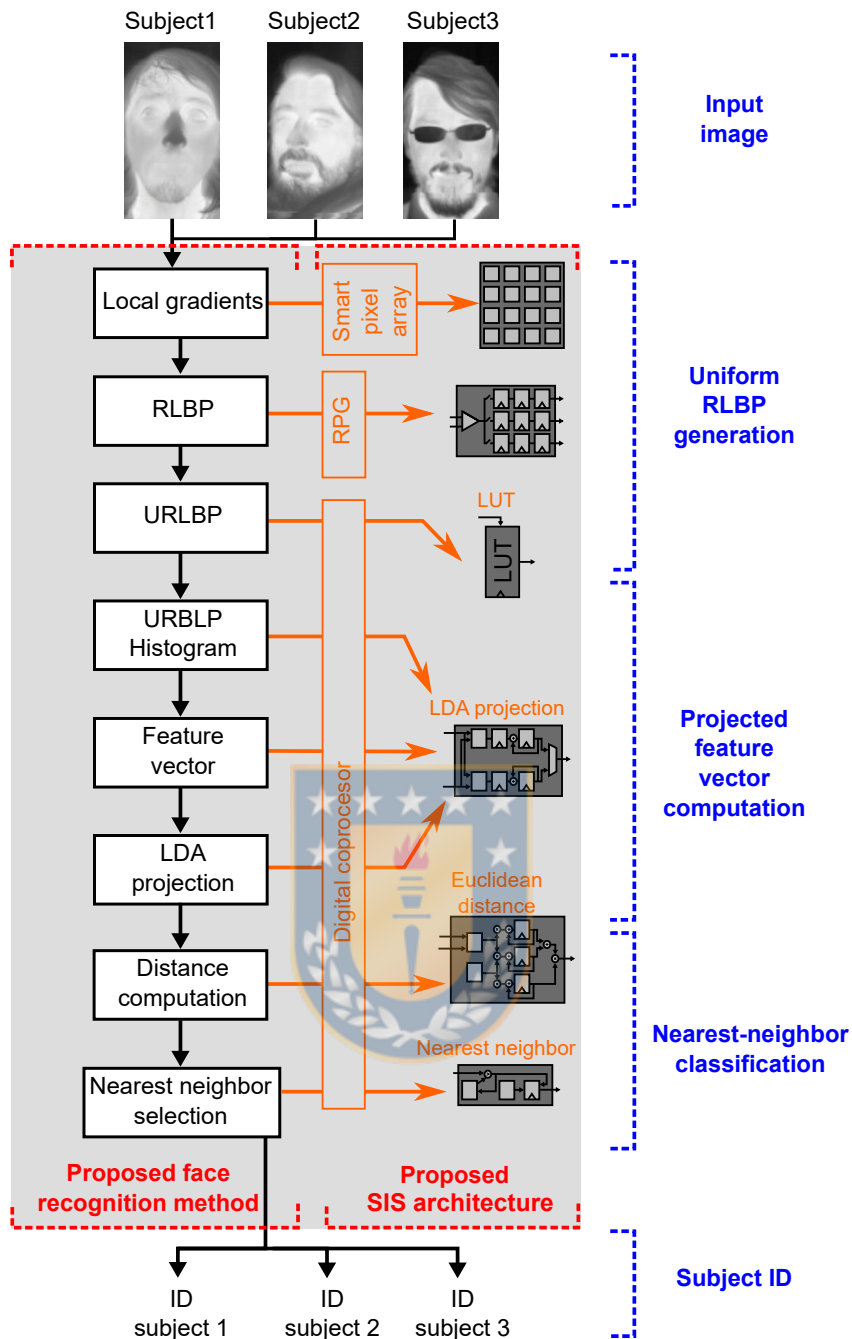


Fig. 3.1: Proposed method and SIS architecture. The left hand side illustrates the steps of the presented face recognition algorithm. The right hand side shows the elements of the proposed SIS, namely smart-pixel array, pattern generator and digital coprocessor, which execute the stages of the algorithm.

pattern for the pixel. Thus, LBP requires 8 comparisons per pixel. In the RLBP method [101], shown in Fig. 3.2b, each pixel is compared only to its rightmost neighbor and the results from the comparisons of the 8 neighbors are concatenated to build the pattern vector. Unlike LBP, RLBP requires only one comparison per pixel, because the result of each comparison is used in 8

Algorithm 1: Proposed method using RLBP + LDA.

input : Input frame $I_{m \times n}$, LDA projection matrix \mathbf{W} , face database FD , number of subjects N , distance threshold THR

output: Subject ID

begin

$U_{m \times n} \leftarrow \text{URLBP}(I)$ using Algorithm 2;

for $i \leftarrow 0$ **to** 7 **do**

for $j \leftarrow 0$ **to** 7 **do**

 Region $R_{i,j} \leftarrow U(\frac{m}{8}i : \frac{m}{8}(i+1) - 1, \frac{n}{8}j : \frac{n}{8}(j+1) - 1)$;

 Histogram $H_{i,j} \leftarrow \text{Histogram}(R_{i,j})$;

Feature vector $X \leftarrow \text{Concatenate} \{H_{0,0} \text{ to } H_{7,7}\}$;

Projected vector $Y \leftarrow \mathbf{W}^T X$;

for $k \leftarrow 1$ **to** N **do**

 Distance $D_k \leftarrow \sqrt{\sum_{i=1}^{N-1} (Y_i - FD_{k,i})^2}$;

if $\min(D_k) > THR$ **then return** *unknown*;

else return *ID of* $\min(D_k)$;

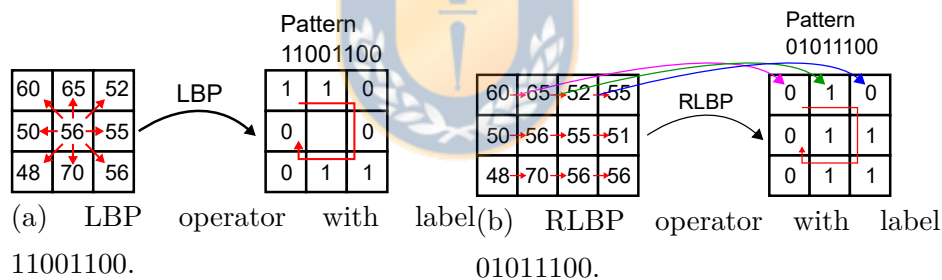


Fig. 3.2: Examples of LBP and RLBP operators on a 3×3 -pixel window.

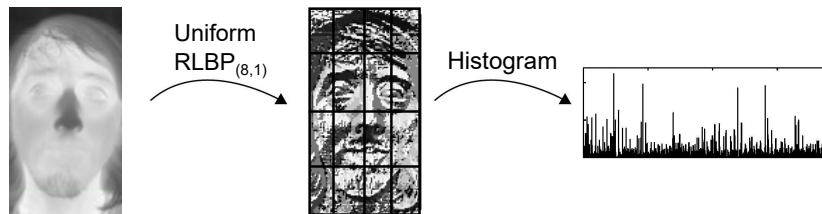


Fig. 3.3: Ahonen's algorithm using uniform RLBP.

different kernels. Moreover, all the comparisons in the image can be performed in parallel using only one comparator per pixel. While the features extracted by RLBP contain less information than LBP, the method provides a sufficiently accurate texture representation of the image that achieves a similar performance in face recognition, as shown in Section 5.2.3.

Algorithm 2: Uniform RLBP computation.

input : Input image $I_{m \times n}$, **LUT** of uniform RLBP values

output: Uniform Ringed LBP **URLBP**
begin

```

for  $I_{i,j} \in I$  do
  if  $I_{i,j} - I_{i+1,j} > 0$  then  $\nabla I_{i,j} \leftarrow 1$ ;
  else  $\nabla I_{i,j} \leftarrow 0$ ;
for  $I_{i,j} \in I$  do
   $RLBP_{i,j} \leftarrow \{\nabla I_{i-1,j-1}, \nabla I_{i,j-1}, \nabla I_{i+1,j-1},$ 
     $\nabla I_{i-1,j}, \nabla I_{i+1,j},$ 
     $\nabla I_{i-1,j+1}, \nabla I_{i,j+1}, \nabla I_{i+1,j+1}\}$ ;
 $URLPB \leftarrow \mathbf{LUT}(RLBP)$ ;
return  $URLPB(X)$ 

```

After computing the binary patterns, the algorithm divides the image into 8×8 nonoverlapping regions, and computes a histogram of the binary patterns in each image, as shown in Fig. 3.3. The 64 resulting histograms are concatenated to produce the feature vector that represents the input image. Ahonen [100] uses uniform LBP to reduce the number of labels in the histogram. Uniform LBP assigns a label only to those patterns that have at most two 0-1 or 1-0 transitions between adjacent positions in the 8-bit pattern. As shown in Algorithm 2, the algorithm uses the same technique, using a 256-entry lookup table (LUT) to map the RLBP values onto uniform RLBP patterns.

After computing the histogram vector, Algorithm 1 uses LDA to map the vector onto a lower-dimensional subspace, as in the Fisherfaces method [102]. LDA applies a linear transformation to the histogram vector, where the transformation matrix \mathbf{W} is computed to minimize the variance between vectors belonging to the same class (images of the same person) and maximize the variance between vectors of different classes. Using LDA allows us to improve the performance of the classifier, use a simple distance metric, and reduce the dimension of the feature vector, reducing the computational complexity of the classifier. The transformation matrix is computed off-line using a labeled training set, and is used to project the histogram vector X onto the new feature space as shown in Eq. (3.1):

$$Y = W^T X \quad (3.1)$$

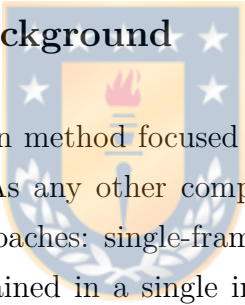
where Y is the linear projection, X is the uniform RLBP histogram vector and W is the LDA

projection matrix.

Finally, in the classification stage, the algorithm computes the Euclidean distance between the projected feature vector and each element of the stored dataset FD of known subjects. FD contains one feature vector for each known face, which is computed as the centroid of all feature vectors obtained from the same subject in the training set. The vectors in FD have also been projected using LDA, and the training set can be the same used to compute the LDA matrix \mathbf{W} . Using the nearest-neighbor criterion, the algorithm labels the input image with the identity of the subject with minimal distance to the projected vector if that distance is larger than a predefined threshold THR . If the minimum distance is larger than THR , the algorithm labels the input image as an unknown subject.

3.2 Object detection

3.2.1 Object detection background



Object detection is a computer vision method focused on detecting instances of visual objects of a class in digital images [103]. As any other computer vision technique, it is possible to classify object detection in two approaches: single-frame and multi-frame. Single-frame object detection uses the information contained in a single image or video frame, to perform shape processing [104,105], color segmentation [105,106], or light intensity segmentation [107]. Multi-frame object detection considers the information of multiple video frames, usually consecutive in time. This allows us to discriminate the background from the foreground.

The most important applications of object detection in the past 20 years include pedestrian detection, face detection, text detection, traffic sign/light detection, and remote sensing target detection [103]. Depending on which application is the detection target, a different approach must be chosen. On applications such as text detection or face detection, static objects are being detected the main concerns are intra-class variation, such as different text fonts [108] or face expressions [109], and object rotation [110] and distortion [111]. Many recent works have used convolutional neural networks (CNNs) to detect objects [103,112], and its variations such as pyramid R-CNN [113], dynamic R-CNN [114], termed oriented R-CNN [115] or libra R-CNN [116]. Despite their very good results, the implementation complexity increases as computational speeds are required to reach real-time processing. Thus, simple methods are still an alternative for fast and low computational cost implementations. One simple object detection

technique is based on object motion, known as motion-based object detection [117,118]. Motion can be estimated in many ways, but always requires storing previous frames to determine whether and object is in movement. The basic motion detection is to consider two consecutive frames and calculate the absolute difference between their respective pixels. Where if that difference is higher than a threshold, then it is possible to assume that those particular pixels are in movement. After detecting objects, it is required to label the pixels or bound them with known shapes to differentiate one of others [119,120], such as bounding boxes or single pixel labeling.

3.2.2 Object detection for SIS

Figure 3.4 shows a block diagram of the proposed object detection algorithm and the hardware modules of the proposed SIS that perform each step of the algorithm. In the center, the figure shows the three core blocks of the SIS, which are the smart pixel array, an analog comparator (A-THR) core and the digital coprocessor. Section 4.2 provides a detailed description of the SIS architecture and circuits.

The proposed algorithm is based on the work by Bir Bhanu et al. [121]. They presented a human motion analysis model in IR video sequences, based in kinematic. To discriminate people from the background, they assume that in a video sequence the only objects in movement are people. With this, it is possible to determine the silhouette of a person by simply subtracting the background from the frame. When the difference of the pixel from the background and the frame surpasses a predefined threshold, then the model considers that pixel as a part of the silhouette. Once the silhouette is extracted from the frame, the model uses the body position and characteristics to determine the 3D human motion for automatic gait recognition.

Algorithm 3 shows the motion-based object detection algorithm, which consist of four main stages: frame-difference, thresholding, morphological operations (erosion and dilation), and connected components. Figure 3.5 illustrates the operation of the algorithm, showing an input image and the output of the first three stages.

The first stage of the algorithm computes the absolute frame-difference: it computes the absolute difference between corresponding pixels in two consecutive video frames. The next stage compares each pixel difference to an application-defined threshold: if the absolute frame-difference is greater than the threshold, the algorithm identifies it as a movement pixel and assign it a logic label of value 1. Otherwise, the pixel is labeled as 0. Figure 3.5(c) illustrates the output

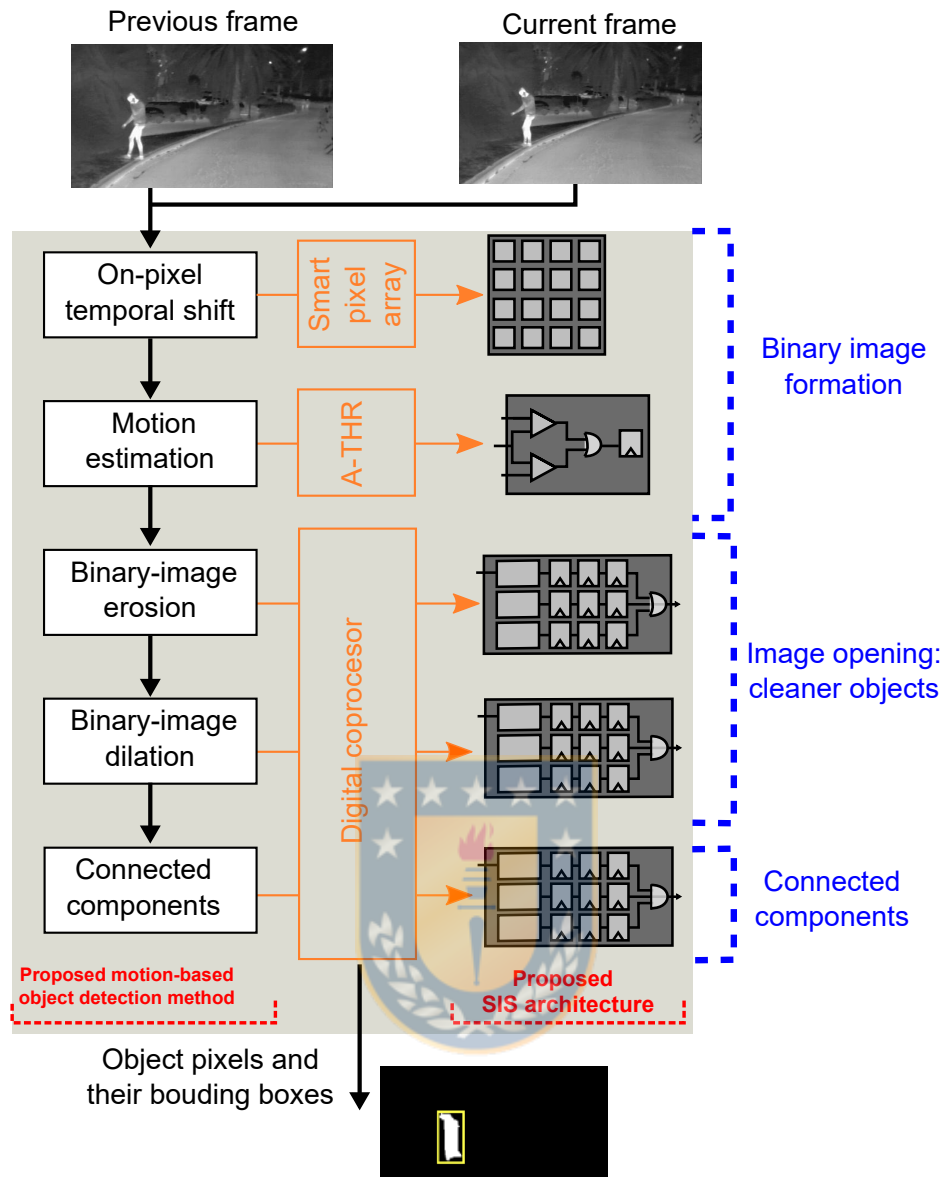


Fig. 3.4: Illustration of the motion-based object detection algorithm and SIS architecture.

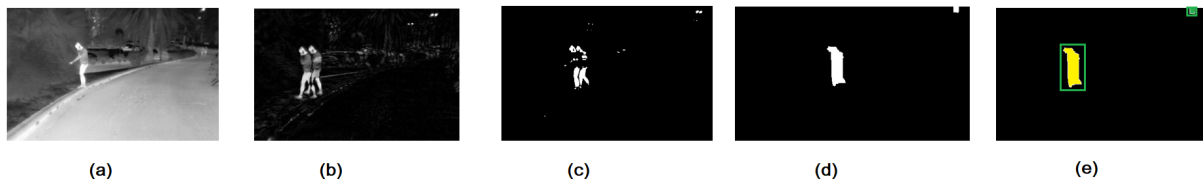


Fig. 3.5: Results of the steps of motion-based object detection: (a) input image, (b) result of the frame-difference, (c) thresholding the frame-difference, (d) applying the image-open morphological transformation, and (e) bounding boxes.

of the threshold stage. The figure shows that this method can produce isolated labels due to abrupt changes in pixel values. To compensate for this, it is common to add a morphological operation stage. The algorithm applies an image opening operation, which consists of image

Algorithm 3: Motion-based object detection.

input : Input frame $im_{k,m \times n}$, previous input frame $im_{k-1,m \times n}$, threshold THR

output: Output frame of highlighted object-pixels and bounding boxes.

begin

for $i \leftarrow 1$ **to** n **do**

for $j \leftarrow 1$ **to** m **do**

 Absolute difference $im_{diff}(i, j) \leftarrow |im_k(i, j) - im_{k-1}(i, j)|$;

 Threshold-image $im_{thr} \leftarrow im_{diff} > THR$;

for $i \leftarrow 1$ **to** n **do**

for $j \leftarrow 1$ **to** m **do**

 Eroded image $im_{erd}(i, j) \leftarrow ERD(im_{thr})$ using Equation (3.2);

for $i \leftarrow 1$ **to** n **do**

for $j \leftarrow 1$ **to** m **do**

 Dilated image $im_{dlt}(i, j) \leftarrow DIL(im_{thr})$ using Equation (3.3);

for $i \leftarrow 1$ **to** n **do**

for $j \leftarrow 1$ **to** m **do**

 Label pixel $im_{lb}(i, j) \leftarrow CC(im_{dlt})$ using Equation (3.4);

 Update bounding boxes;

return Image with highlighted objects im_{dlt} , labeled objects im_{lb} and bounding boxes

erosion followed by dilation. Figure 3.5(d) shows that this considerably reduces the number of isolated labels. Finally, Fig 3.5(e) shows, using different colors and bounding boxes, the output of the connected components algorithm, which labels the objects found in the image. From this point, it is possible to further extend image analysis to process shapes, single objects, and more.

The implemented morphological image transformation uses 3×3 -pixel kernels, for both image erosion and dilation. Since the algorithm applies morphological transformation to binary images, the calculation can be simplified.

Figure 3.6a depicts image erosion, which replaces the center pixel in a 3×3 window with the minimum value in the window. As shown in Figure 3.6b with binary images, dilation replaces the pixel with logical 0 if there is at least one pixel equal to 0 in the window, as described in Eq. (3.2):

$$im_{erd} = ERD(im_{in}) = \begin{cases} 0 & \text{if at least one pixel is 0} \\ 1 & \text{if all pixels in kernel are 1} \end{cases} \quad (3.2)$$

7	8	8	9	10
14	15	7	6	1
6	7	4	6	3
11	8	3	7	8
11	11	9	11	13

→

7	8	8	9	10
14	15	7	6	1
6	7	3	6	3
11	8	3	7	8
11	11	9	11	13

(a) Conventional image erosion.

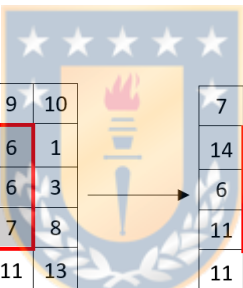
0	0	0	1	1
1	0	1	1	0
0	0	1	1	1
1	0	0	1	0
1	1	0	1	1

→

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

(b) Binary image erosion.

Fig. 3.6: Graphic description of (a) conventional image erosion and (b) binary image erosion. Conventional image erosion replaces central pixel with the lower value on its neighborhood. Binary image erosion replaces central pixel with a logical 0 if there is at least one logical 0 on its neighborhood.



7	8	8	9	10
14	15	7	6	1
6	7	4	6	3
11	8	3	7	8
11	11	9	11	13

→

7	8	8	9	10
14	15	7	6	1
6	7	15	6	3
11	8	3	7	8
11	11	9	11	13

(a) Conventional image dilation.

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

→

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

(b) Binary image dilation.

Fig. 3.7: Graphic description of (a) conventional image dilation and (b) binary image dilation. Conventional image dilation replaces central pixel with the higher value on its neighborhood. Binary image dilation replaces central pixel with a logical 1 if there is at least one logical 1 on its neighborhood.

Figure 3.7a depicts image dilation, which replaces the center pixel in a 3×3 window with the maximum value in the window. As shown in Figure 3.7b with binary images, dilation replaces

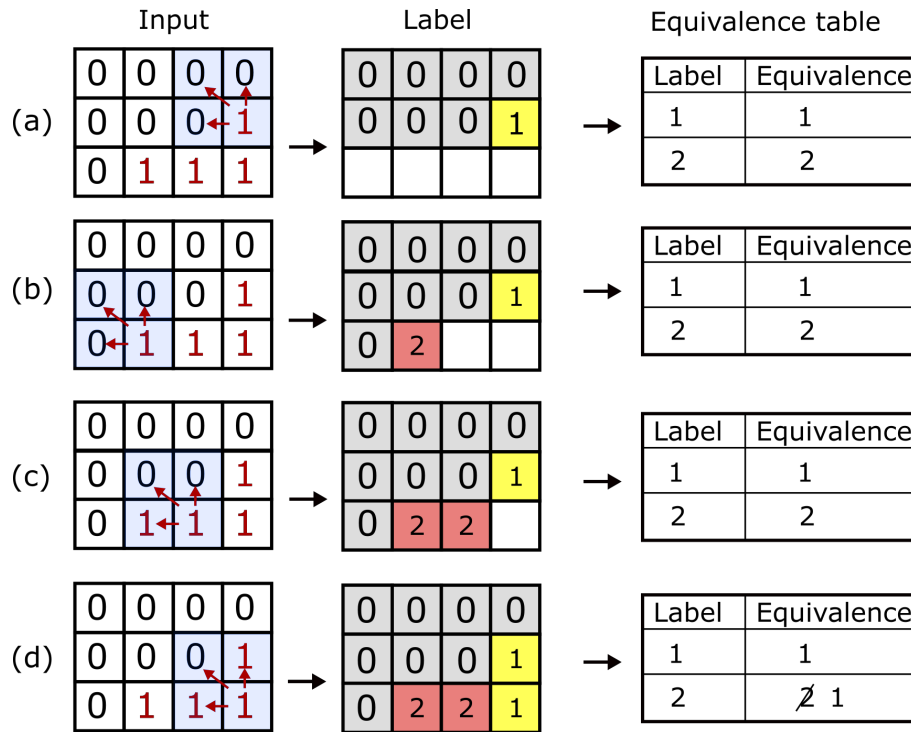


Fig. 3.8: Graphic description of the connected components algorithm. The algorithm receives as input a binary image with movement pixels from the frame-difference stage. The algorithm defines a 2×2 -pixel window, corresponding to the analyzed pixel, and its west, northwest and north neighbors. For each pixel, if its three neighbors are not labeled as part of an object, then the algorithm assigns a new label to the pixel (Figs. a and b). If one or more of the neighbors have the same label, the algorithm assigns the same label to the pixel (Fig. c). When two or more neighbors have different labels, the algorithm assigns one of the labels to the pixel and update the equivalence table to reflect that all those labels now belong to the same connected component (Fig. d).

the pixel with logical 1 if there is at least one 1 pixel in the window, as described in Eq. (3.3):

$$im_{dil} = DIL(im_{in}) = \begin{cases} 0 & \text{if all pixels in kernel are 0} \\ 1 & \text{if at least one pixel is 1} \end{cases} \quad (3.3)$$

After computing the opening operation, the connected components algorithm assigns all connected pixels to the same object in the image and computes the bounding box for each detected object. The connected components block operates in a single pass and outputs a table containing the bounding boxes for all objects in the image.

The connected-components algorithm finds the objects in an image by analyzing the move-

ment pixels and their adjacent neighbors. Figure 3.8 illustrates the operation of the algorithm. For each movement pixel in the image, the algorithm looks at its north, northwest and west neighbors. If none of them are also a movement pixel, the algorithm assigns a new object label to the pixel, as shown in Figs. 3.8(a) and (b). Otherwise, if the neighbor movement pixels are part of the same connected component, the algorithm assigns the same object label to the new pixel, thus adding it to the connected component (Fig. 3.8(c)). If the neighbor movement pixels belongs to different connected components, the algorithm assigns one of the labels to the new pixel and merges the connected components by adding a new entry into the equivalence table (Fig. 3.8(d)). The base procedure of the algorithm is described as the priority-OR operation in Eq. (3.4):

$$im_{tb} = CC(im_{in}) = \begin{cases} L_0 & \text{if } p_s \text{ is 0, or} \\ L_{i+1} & \text{if only central pixel is 1} \\ L_{nw} & \text{if central and north-west pixels are 1} \\ L_n & \text{if central and north pixels are 1} \\ L_w & \text{if central and west pixels are 1} \end{cases} \quad (3.4)$$

where L_0 is the label for no-object pixels, and L_w , L_{nw} , and L_n are labels of the west, northwest, and north pixels, respectively.

Every time the algorithm creates a new connected component or adds a pixel to an existing component, it updates the coordinates of its bounding box in a table. When the algorithm merges two connected components, it updates the bounding box .

4. SIS architectures

4.1 SIS architecture for face recognition

Figure 4.1 shows the proposed SIS architecture for face recognition, which can be configured to operate as a conventional image sensor or as a face recognition system. The main components are an array of smart pixels, an Ringed Local Binary Pattern (RLBP) generator (RPG), and a digital coprocessor. The pixel array acquires image data and, in parallel, subtracts the values of horizontally adjacent pixels. The row-select and column-select circuits sequentially read the pixel values and send them to the RPG, which constructs an 8-bit RLBP for each pixel in the image. Dividing the image into 8×8 regions, the digital coprocessor computes a histogram of uniform RLBP for each region and concatenates them to form the feature vector. Then, it projects the vector using LDA, computes its Euclidean distance to a set of stored vectors corresponding to the known faces, and labels the image using a nearest-neighbor criterion.

The SIS can also output a conventional image, in which case each smart pixel outputs the analog-voltage output of its readout circuit, and the row- and column-select circuits read the voltages to an ADC that outputs the digital value of the pixels.

4.1.1 Smart pixel

Figure 4.2 shows the circuit that implements the smart pixel. It consists of a photodetector, a pair of input-select switches, a programmable CTIA, and a row-select switch. The input of the CTIA are the currents from the local or horizontally-adjacent pixel, selected by the input-select switches. The CTIA computes a voltage that represents either the current pixel value or the difference between adjacent pixels, configured by the global control lines NegInt and PosInt.

Although it uses more area than alternative pixel circuits, using a CTIA for photocurrent integration is a preferred method for low-light environments and IR cameras [122–125] because its low input impedance offers good injection efficiency with weak photodiode currents. In particular, as discussed in Section 5.2.3, this work is interested in using the smart pixel to recognize faces in thermal IR video. Moreover, when compared to other pixel circuits, a CTIA features a wide linear output voltage range [126], small frame-to-frame lag, and reduced noise

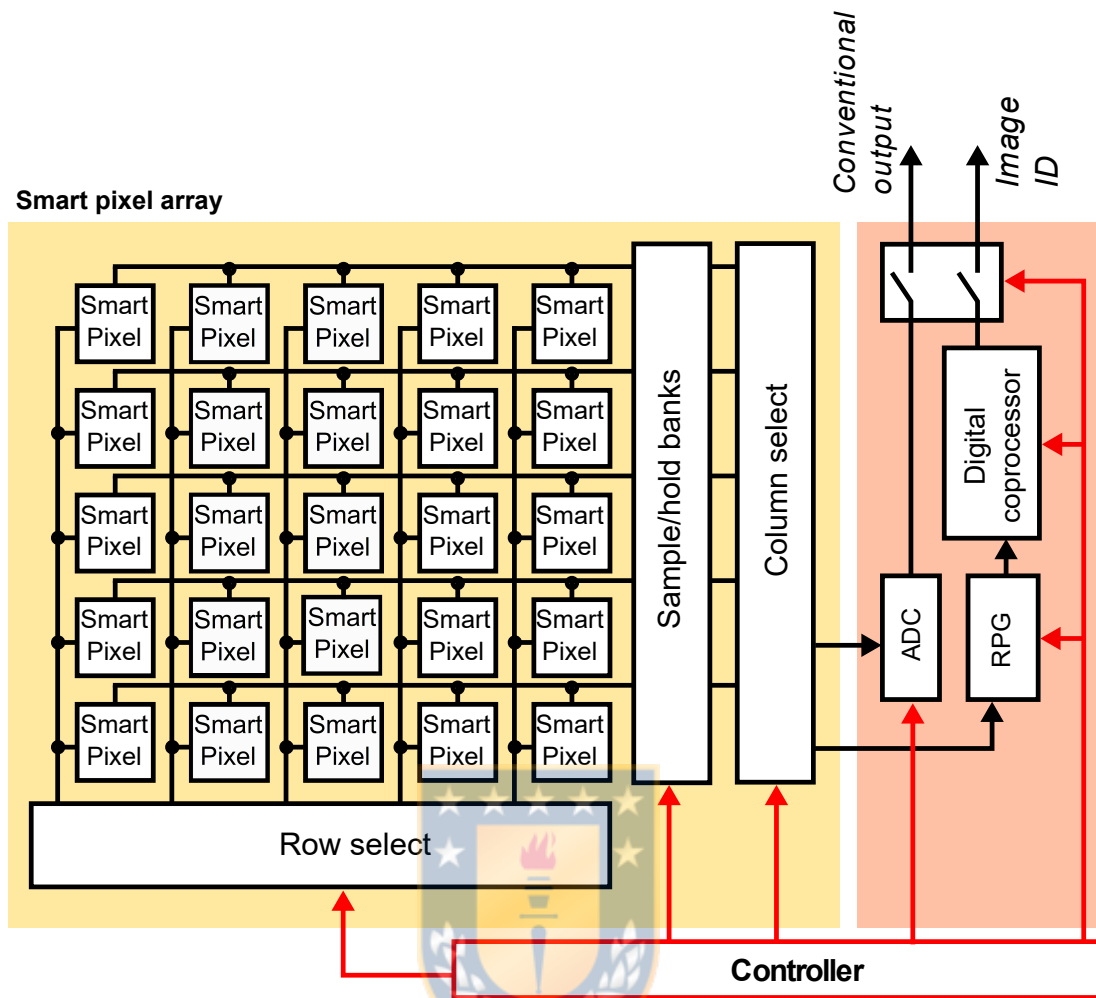


Fig. 4.1: Architecture of the proposed SIS. An array of smart pixels outputs either the pixel value or the difference between horizontally adjacent pixels. An RLBP generator (RPG) reads pixel values and creates an 8-bit RLBP for each pixel in the image. . The digital coprocessor computes histograms of RLBP patterns to construct the feature vector, executes the LDA projection on each vector, and selects the nearest neighbor from a stored set of projected vectors using the Euclidean distance.

through better control of the photodiode bias [127].

Figure 4.3 shows the schematic of the CTIA. It integrates its input current to produce an output voltage, and a set of 4 switches, implemented as conventional CMOS transmission gates, can control the orientation of the integration capacitor [128]. The input current comes from the photodetectors in the local or adjacent pixel.

Figure 4.4a shows the CTIA operating in conventional mode. During the entire integration time, input-control switch connects the CTIA input to the local photodetector PD1. The CTIA is configured in direct mode: $sw1$ and $sw4$ are closed, and $sw2$ and $sw3$ are open. The equivalent

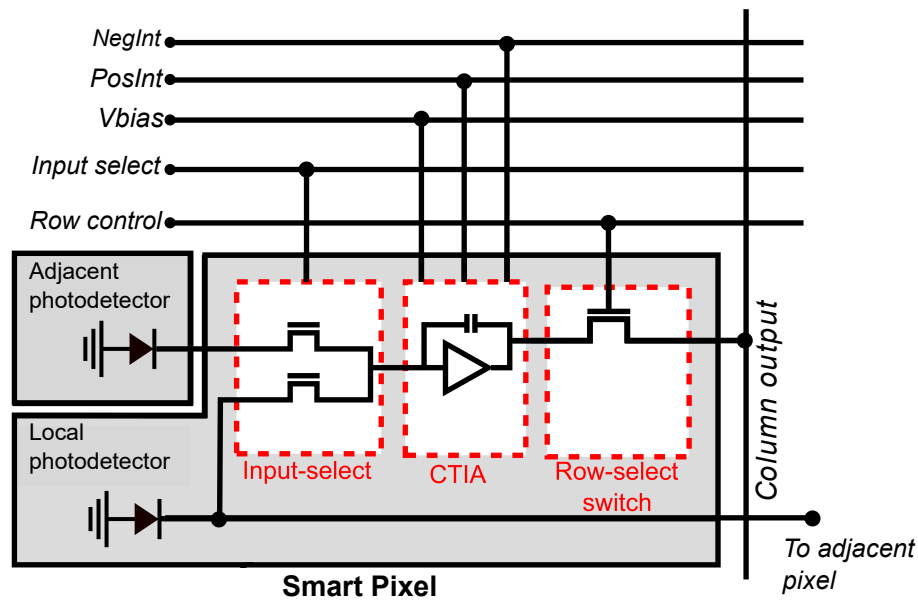


Fig. 4.2: The smart pixel consists of an analog input-select multiplexer, a configurable CTIA, and a row-select switch. All the smart pixels in the array share the control signals placed above in the figure, and all the pixels in the same column share the *Column output* signal. The input to the CTIA can be selected from the photodetector in the local or adjacent pixel.

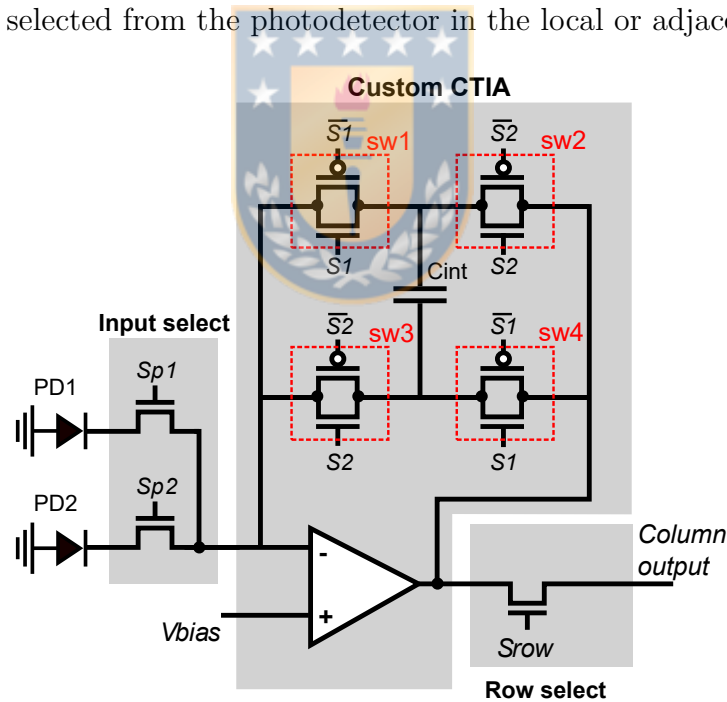


Fig. 4.3: Schematic diagram of the configurable CTIA. The CTIA integrates the photodetector currents and outputs a voltage that represents either the pixel value or the difference between horizontally-adjacent pixels.

circuit of Fig. 4.4b shows the CTIA acting as a conventional integrator, and Eq. (4.1) shows how it computes the output voltage that represents the local pixel value:

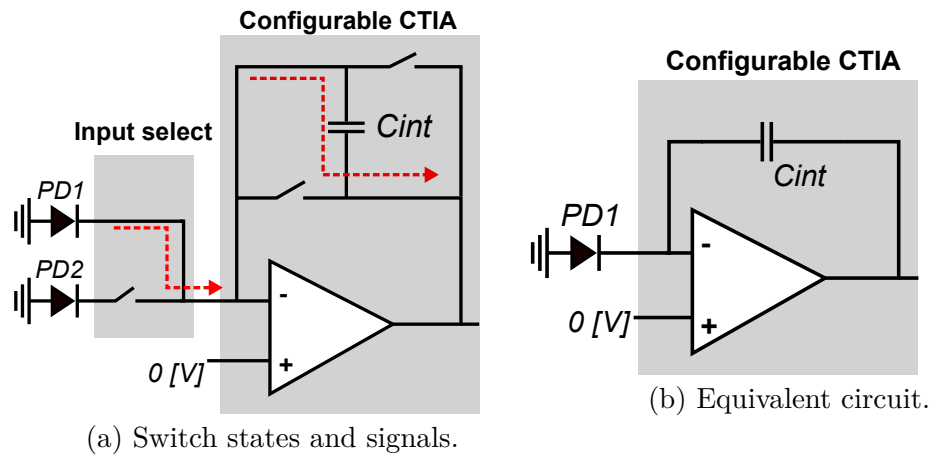


Fig. 4.4: Smart pixel in conventional mode: the input-select switches pass the current from PD1, $sw1$ and $sw4$ are closed to integrate the current, and $sw2$ and $sw3$ stay open.

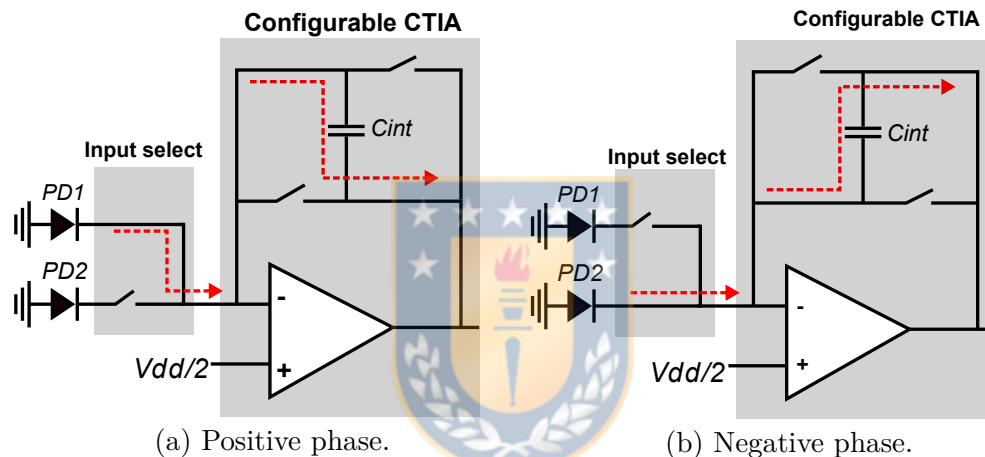


Fig. 4.5: Simplified view of positive and negative integration. During positive integration: $sw1$ and $sw4$ stay closed, $sw2$ and $sw3$ stay open. During negative integration: $sw2$ and $sw3$ stay closed, $sw1$ and $sw4$ stay open.

$$V = I\Delta t/C_{int}, \quad (4.1)$$

where V is the output voltage, I is the input current from photodetector PD1, Δt is the integration time, and C_{int} is the capacitance value.

Figure 4.5 shows the smart pixel when configured to compute local horizontal gradients. The global bias input of the CTIA is set to the midpoint between the rails (1.65 V for a 3.3 V supply voltage). The integration time is divided into two phases of equal duration: direct and inverse. During the direct phase, shown in Fig. 4.5a, the CTIA operates in conventional mode by integrating the current from the local PD1 detector, starting from 1.65V. During the negative phase, shown in Fig. 4.5b, the input switches select the current from the local neighbor

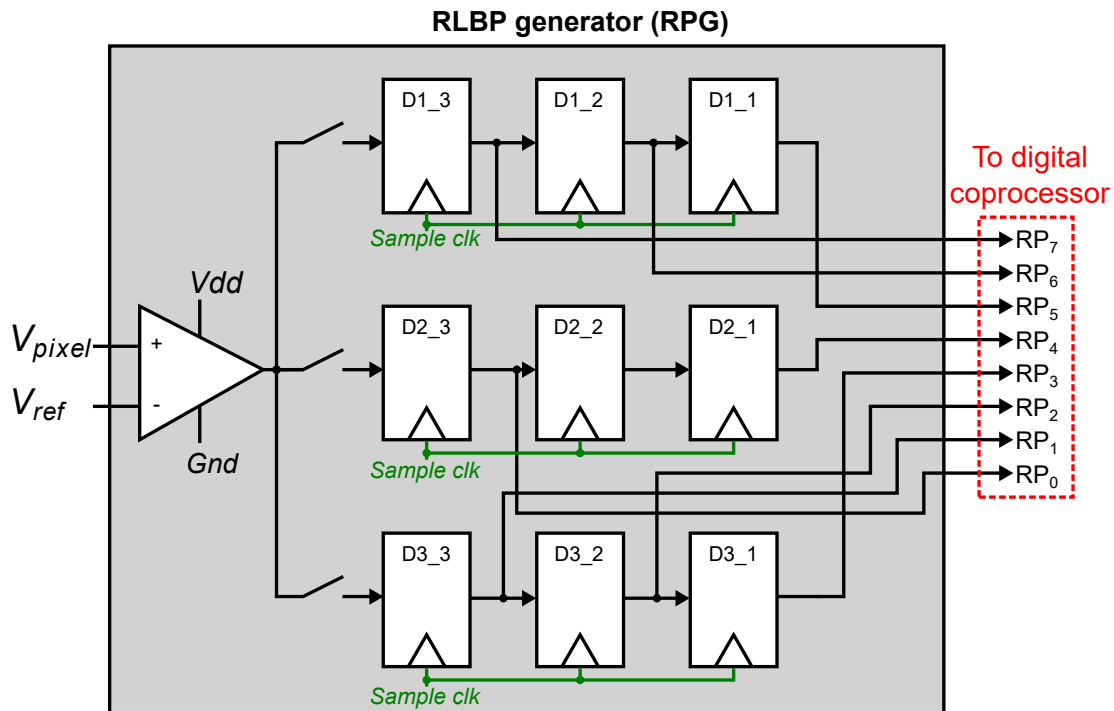


Fig. 4.6: Architecture of the RPG. An input comparator compares the local gradient value for each pixel to a reference voltage. The digital comparator outputs are sequentially stored in an array of 3×3 flip-flops, organized as 3 shift registers. The RPG outputs an 8-bit RLBP with the output of all the flip-flops except for the one at the center.

pixel PD2, $sw1$ and $sw4$ are open, and $sw2$ and $sw3$ are closed. Therefore, during the inverse phase, the CTIA integrates the negative current value of the PD2 photodetector. The output voltage at the end of the integration period is computed as shown in Eq. (4.2):

$$V = (I_1 \Delta t_s + I_2 \Delta t_s) / (2C_{int}), \quad (4.2)$$

where V is the output voltage, I_1 is the input current from the local detector PD1, I_2 is the current from the adjacent detector PD2, Δt_s is the integration time, and C_{int} is the capacitance.

In local-gradient mode, the integration time per pixel is reduced by 50% compared to normal operation, which decreases the signal-to-noise ratio. However, this allows us to compute local spatial gradients in parallel on the entire FPA with very small area overhead compared to a conventional integrator. These local gradients are then used by the RPG to compute the RLBP for each pixel.

4.1.2 RLBP generator

Figure 4.6 shows the topology of the RPG circuit. An input opamp compares the readout value V_{pixel} , which represents the difference between two adjacent pixels, to a global reference voltage V_{ref} . When $V_{pixel} > V_{ref}$, the digital output of the comparator is 1, and 0 otherwise. The output of the comparator is written into an array of 3×3 flip-flops configured as three shift registers, which is used to create the RLBP.

To compute the RLBP in each region, the RPG performs a row-wise read of the FPA. For each pixel, the RPG reads the pixel value and its two vertically-adjacent neighbors. The comparator output for these values is written into flip-flops D1_3, D3_3 and D3_3. The register array then performs a right shift, and the next three pixels are read from the FPA. When nine reads have been completed, the array holds the RLPB for the central pixel, which is then sent to the digital coprocessor to compute the histogram. Because the 3×3 -pixel windows used to compute the RLBP overlap for adjacent pixels, the next RLBP is completed after three reads. The process continues until all pixel values in the region have been read, and the RPG moves to the next region in the image (as shown in Fig. 3.3).

Because the FPA directly outputs the local pixel differences, computing the RLBP requires only a 3×3 -bit array instead of the large line buffers that would be required to compute the differences in the RPG. Each RLBP requires three reads from the FPA, but because these reads are only used for a 1-bit comparison instead of a complete analog-to-digital conversion, these reads complete significantly faster than when the array operates in conventional mode.

4.1.3 Digital coprocessor

The digital coprocessor is responsible for computing the histograms of RLBP from the image, normalizing and centering the data, projecting the resulting histogram vector using LDA, computing the Euclidean distance between the projected vector and a stored database of known faces, and selecting a label for the input image using a nearest-neighbor criterion.

Figure 4.7 shows the architecture of the face-recognition coprocessor. It receives as input the 8-bit RLBP vector RP from the RPG module. The memory controller reads the LDA coefficients from external RAM and sends them to the LDA projection module. This module reads the patterns computed by the RPG for each region of the image, and computes the histogram vector and projects it using the LDA coefficients. Histogram computation and LDA

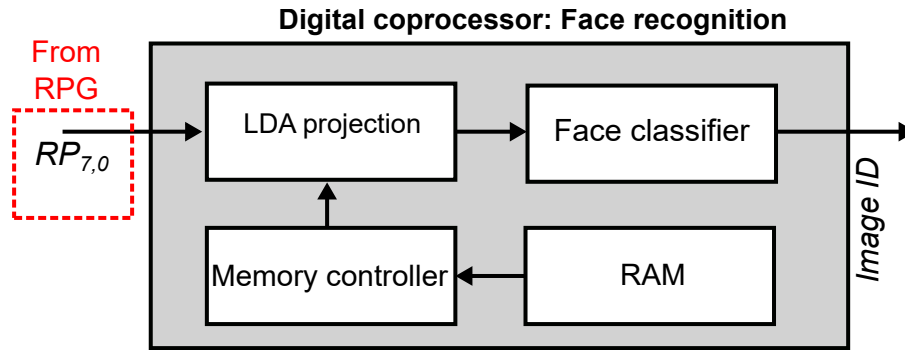


Fig. 4.7: Architecture of the digital coprocessor. The processor receives a stream of RLBP's from the RPG, and simultaneously builds the histogram vector and projects it using LDA. A memory controller retrieves the LDA coefficients from RAM. The Euclidean distance between the projected vector and the contents of database of stored faces is used for classification with the nearest-neighbor criterion.

projection are fused into a single step to save memory and arithmetic resources. The output of the LDA module is the feature vector of the input image projected onto the LDA subspace. The face recognition module computes the Euclidean distance between this vector and a set of stored vectors that represent the known faces. The module selects the minimal distance and compares it against a chosen threshold. When the distance is smaller than the threshold, the module outputs the ID of the selected known face. Otherwise, it outputs a null value.

Figure 4.8 shows the architecture of the LDA projection module. The module receives a stream of 8-bit RLBP's from each region of the image. The first step converts the RLBP into a 6-bit uniform RLBP (uRP) using a 256-entry lookup table. In order to avoid the use of multipliers and reduce the amount of local storage required by the LDA projection, the module computes the histogram vector and the multiplication by the projection matrix W simultaneously. Each uRP value denotes a position in the histogram vector, for the current region, that must be incremented to build the histogram. The final value stored in this position should then be multiplied by the corresponding set of coefficient values in W when the vector is multiplied by the matrix. Instead, every time a new uRP is received, the module obtains the value of the coefficient associated with the uRP using a 64-entry coefficient buffer and accumulate the values of the coefficients to directly produce an element of the projected vector. For illustration purposes, let us assume a histogram vector of size three and a projection matrix of size 2×3 . The traditional projection is computed as shown in Eq. (4.3):

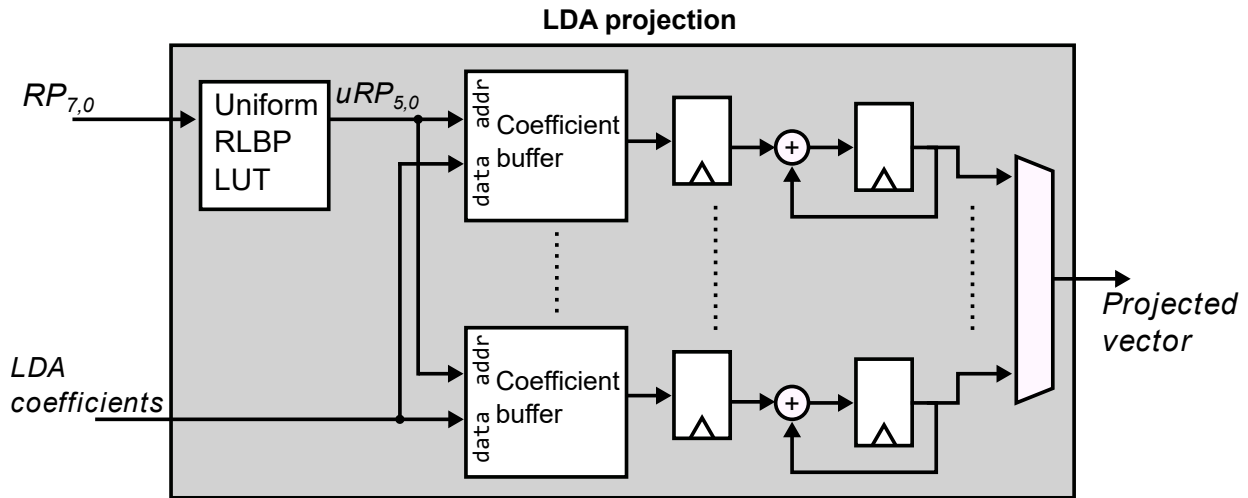


Fig. 4.8: Architecture of the LDA projection module. The module transforms the 8-bit RLBP into a 6-bit uniform RLBP (uRP). For each uRP value received, the module accumulates the value of its corresponding LDA coefficient, thus performing histogram computation and LDA projection in a single step.

$$y = W^T x = \begin{pmatrix} w_{1,1} & w_{1,2} & w_{1,3} \\ w_{2,1} & w_{2,2} & w_{2,3} \end{pmatrix} \cdot \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} w_{1,1}x_1 + w_{1,2}x_2 + w_{1,3}x_3 \\ w_{2,1}x_1 + w_{2,2}x_2 + w_{2,3}x_3 \end{pmatrix} \quad (4.3)$$

where W is the LDA projection matrix, x is the histogram vector, and y is the projected feature vector. Instead, when the LDA module receives the uRP pattern 1, it retrieves the coefficients $w_{1,1}$ and $w_{2,1}$ from the two coefficient buffers in Fig. 4.8, and accumulates these values in the corresponding registers in the figure. When the module receives the uRP pattern 2, it accumulates the coefficient values $w_{1,2}$ and $w_{2,2}$. If the uRP value is 3, the module adds the values $w_{1,2}$ and $w_{2,2}$ to the registers. When all the uRP values have been read, the registers store the coefficient values of the projected vectors. Thus, for a $n \times m$ coefficient matrix, the LDA module requires n coefficient buffers of m elements. The memory controller block is responsible for reading the coefficient values stored in external RAM storing them in the coefficient buffers.

Projecting the histogram vector x with LDA requires normalizing and centering the value of x . Because the centering operation is linear, it can be performed in the projected subspace to reduce the arithmetic hardware required to perform the operation, as shown in Eq. (4.4):

$$y = \eta W^T (x - \mu) = \eta W^T x - \eta W^T \mu, \quad (4.4)$$

where W^T is the LDA projection matrix, η is the scalar normalization coefficient and μ is the

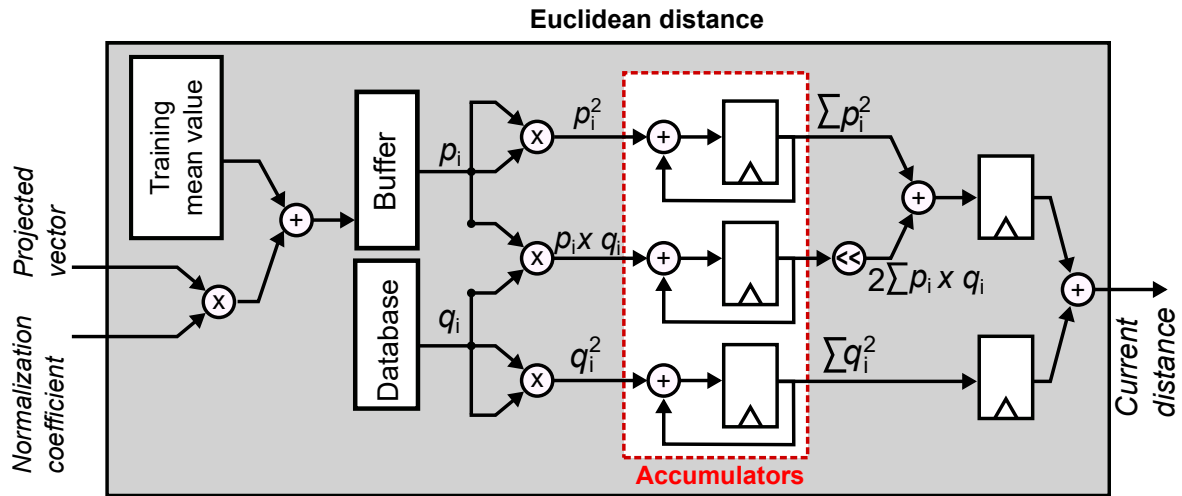


Fig. 4.9: Euclidean distance module. It normalizes and centers the input vector and computes the distance between vectors p and q as $\sum p_i^2 - 2 \sum p_i q_i + \sum q_i^2$.

mean value of the training data vectors. The module performs these operations in the projected subspace, locally storing the value of η and the precomputed value of $\eta W^T \mu$.

The Euclidean distance between two vectors p and q can be computed as shown in Eq. (4.5):

$$d(p, q) = \sqrt{\sum_{i=1}^N (p_i - q_i)^2} = \sqrt{\sum p_i^2 - 2 \sum p_i q_i + \sum q_i^2}, \quad (4.5)$$

where $d(p, q)$ represents the Euclidean distance between vectors p and q , p_i and q_i are the i^{th} components of vectors p and q , respectively, and N is the dimension of the vectors. Because the proposed work is only interested in determining the vector q in the database that is closest to the projected input vector p , it is possible to use the square of the distance $d(p, q)^2$ and avoid computing the square root.

Figure 4.9 shows the architecture of the Euclidean distance module, which computes $\sum p_i^2 - 2 \sum p_i q_i + \sum q_i^2$. The inputs to the module are the projected vector and the LDA normalization coefficient. As described above, the input vector is normalized and centered in the LDA projected space, and stored into a local buffer. Then, the module sequentially computes the distance between the input vector p and each vector q in the database of known faces. It first computes p_i^2 , $p_i q_i$ and q_i^2 , and accumulates their values for $1 \leq i \leq N$ in three registers. Finally, the values of the registers are added in a two-stage pipeline to compute $d(p, q)$, where the value of $\sum p_i q_i$ is shifted one bit to the left to multiply it by 2. The process is repeated for each know-face vector stored in the local database.

Figure 4.10 shows the final stage of the classifier in the digital coprocessor. The module

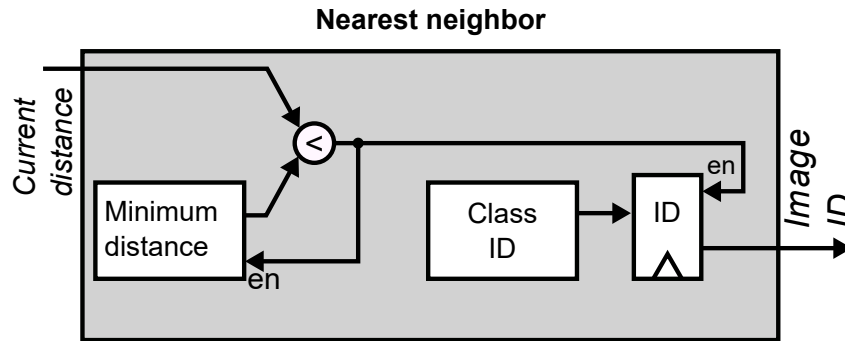


Fig. 4.10: Classification module. The module implements a nearest-neighbor criterion by selecting the face label that corresponds to the minimum distance computed between the input image and the stored database of know faces.

receives a sequence of distances between the input image and each projected vector stored in the database of known faces. It sequentially computes the minimum value of these distances by comparing the currently-stored distance to the incoming value, and updating the register with the smallest value. Finally, the minimum value is compared to a user-supplied threshold. If the value is smaller than the threshold, the module outputs the face label corresponding to the stored minimum value. Otherwise, it outputs a zero to indicate that the input face is not in the database.

4.2 SIS architecture for motion-based object detection

Fig. 4.11 shows the architecture of the proposed SIS. The SIS supports two operation modes: standard imager and motion-based object detection. The core blocks are an array of smart pixels, an analog comparator (A-THR), and a digital coprocessor. In the standard mode, the pixel array acquires image data as a conventional image sensor, row- and column-select circuits sequentially read the pixel data, and an ADC produces digital values as a pixel stream. In object detection mode, the array acquires and stores the image data for the current frame, and computes the difference between the current frame and the stored data for the previous frame. The row- and column-select circuits sequentially read the frame differences and send them to the A-THR, which determines if the absolute difference of each pixel in the image is greater than the application-defined threshold. The output of the A-THR is a single bit for each pixel, that serves as input to the digital coprocessor. The coprocessor computes the binary erosion and dilation, the connected components, and outputs the bounding boxes and binary image.

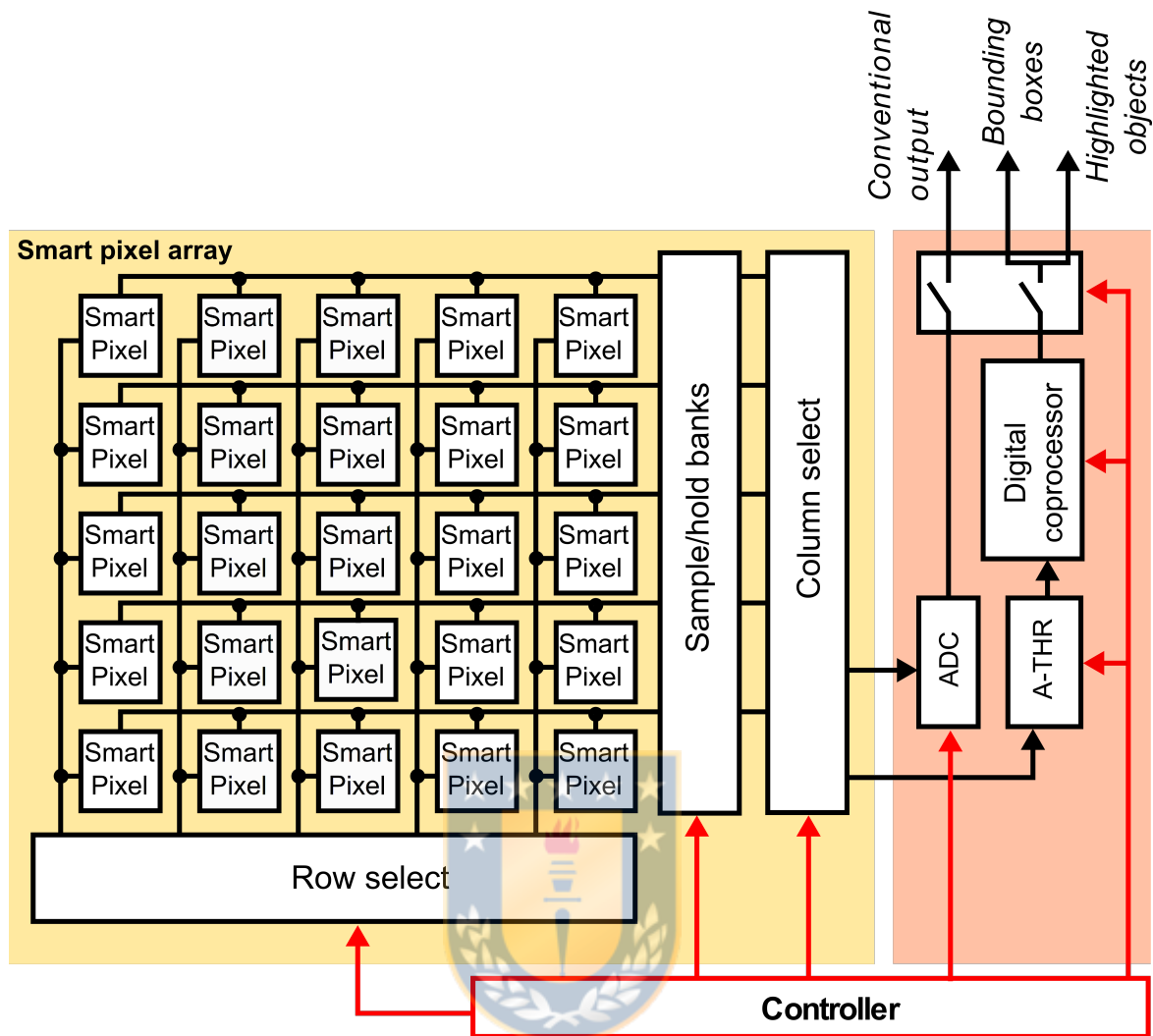


Fig. 4.11: Architecture of the proposed SIS. An array of smart pixels outputs either the pixel value or frame-difference. The A-THR module determines whether the absolute value of the frame-differences exceeds an application-defined threshold. The digital coprocessor computes image opening to improve the object detection, and uses a connected components algorithm to detect objects in the image and compute their bounding boxes. The digital coprocessor can be configured to output the original image or the binary image and the bounding boxes for the objects.

4.2.1 Smart pixel

Figure 4.12 shows a block diagram of the proposed smart pixel composed of a photodetector, switches for input enable and row-select, and a programmable CTIA. The input of the CTIA is connected to a photodetector. The CTIA uses the signals *NegInt*, *PosInt* and *BuffSL* to integrate the photodetector current. The resulting voltage represents the value of the pixel in a frame or the difference between the pixel values in the current and past frames.

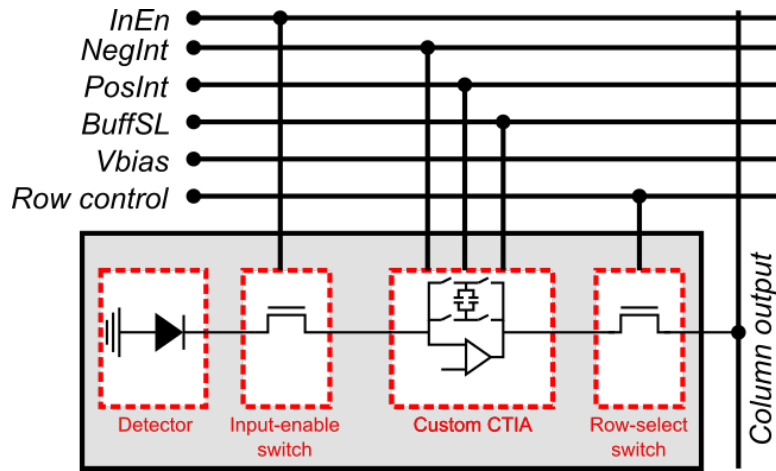


Fig. 4.12: Architecture of the smart pixel. $NegInt$, $PostInt$, $BuffSL$ and $Vbias$ are global bias and control signals. $Row\ control$ is shared by all the pixels in a row and $Column\ output$ is shared by all the pixels in the same column.

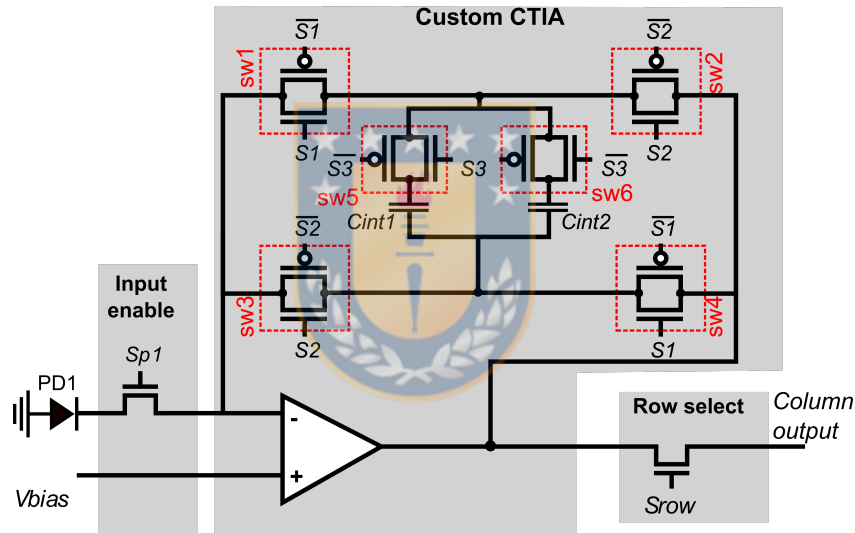


Fig. 4.13: Configurable CTIA. The output voltage of the CTIA represents either the pixel value or the difference between the pixels in the current and past frame. The configurable CTIA includes two integration capacitors C_{int1} and C_{int2} of equal size, which are used as double buffers to integrate and compute the frame difference.

A schematic view of the smart pixel is shown in Fig. 4.13. The CTIA includes two identical integration capacitors that operate as double buffer, and 6 CMOS switches based on conventional transmission gates. The switches are controlled by three configuration signals and their complements. The configurable CTIA integrates its input current to produce an output voltage and uses the switches to select the integration capacitor and control the direction of the integration [64, 66].

The operation in conventional mode of the smart pixel is shown in Fig 4.14a, also referred

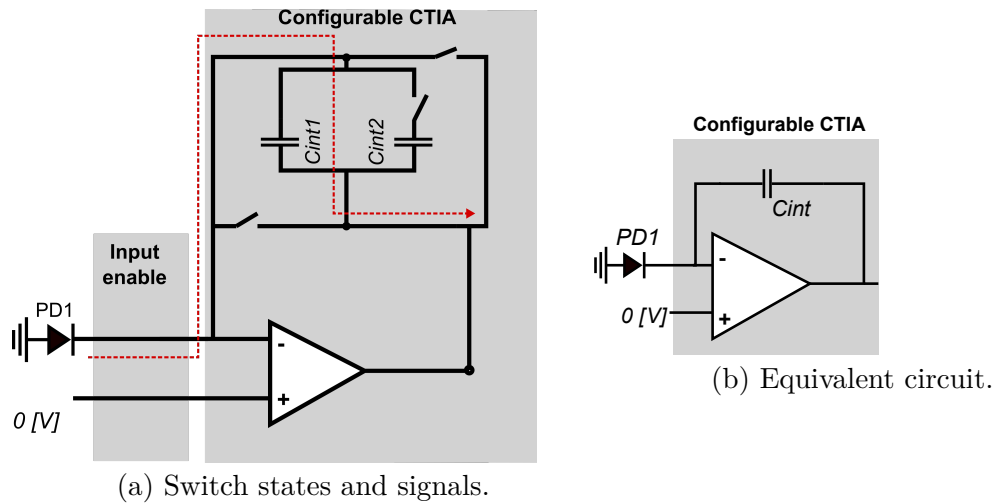


Fig. 4.14: Smart pixel in conventional mode: the input-enable switch passes the current from the photodiode PD1, $sw1$, $sw4$ and $sw5$ are closed to integrate the current using C_{int1} , and $sw2$, $sw3$ and $sw6$ stay open.

to as direct mode. In conventional mode the operation of the smart pixel is equivalent to the conventional CTIA. Similarly, as described in Section 4.1.1, the smart pixel sets the bias voltage to $0V$, and the CTIA integrates the input current on the capacitor C_{int1} . As shown in Fig. 4.14b, in direct mode the smart pixel works as a conventional CTIA, where it sets the switches $sw1$, $sw4$ and $sw5$ as closed, and $sw2$, $sw3$ and $sw6$ as opened. Equation (4.6) describes the output value at the end of the integration time:

$$V = I\Delta t/C_{int1}, \quad (4.6)$$

where V is the voltage at the output of the smart pixel, I is current from photodetector PD1, Δt is the amount of time that takes to integrate, and C_{int1} is the capacitor value.

The operation of the smart pixel when computing the frame-difference in the pixel is shown in Fig. 4.15. The smart pixel sets the global bias input the midpoint of the operation voltage. During a single video frame, the circuit operates in two stages: store and subtract, assigning one half of the integration time to each. During the store stage, the circuit integrates the input current into one of the capacitors, which will be used in the next video frame. During the subtract phase, the CTIA subtracts the input current from the second capacitor, which stores the pixel value of the previous frame. In the next video frame, the capacitors are switched.

Figures 4.15a and 4.15b show the equivalent circuits during an odd video frame. Here, the store and subtract phases integrate the input current in the positive direction in both capacitors: $sw2$ and $sw3$ are closed, and $sw1$ and $sw4$ are open. During the store phase, shown

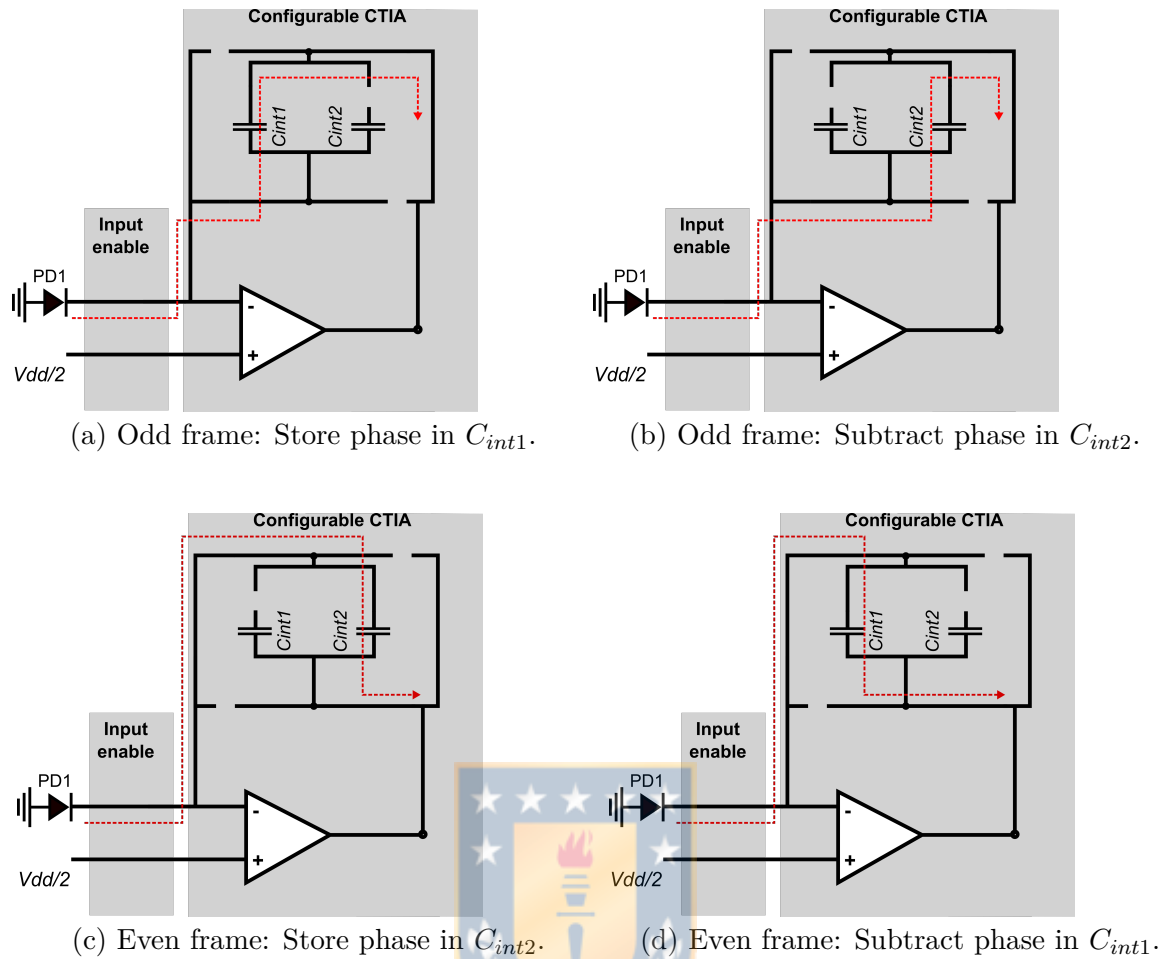


Fig. 4.15: Simplified view of the CTIA in frame-difference mode during odd and even frames. During an odd frame, $sw2$ and $sw3$ are closed while $sw1$ and $sw4$ are open. During the store phase, $sw5$ is open and $sw6$ is closed, and during the subtract phase the states of $sw5$ and $sw6$ are reversed. At the end of the frame, the voltage across C_{int1} represents the frame-difference between the current and previous frame. During even frames the state of all switches is the complement of the odd frames, and the frame-difference is represented by the voltage across C_{int2} .

in Figure 4.15a, $sw5$ is closed and $sw6$ is open to integrate the input current on C_{int1} . During the subtract phase, shown in Figure 4.15b, $sw6$ is open and $sw5$ is closed to integrate (subtract) the input current on C_{int2} , which contains the pixel value acquired in the previous frame value. At the end of the frame time, the voltage across C_{int1} represents the current pixel value to be used in the next frame and the voltage across C_{int2} is the frame-difference between the current and previous frames.

Figures 4.15c and 4.15d show the equivalent circuits during an even video frame, which integrate the input current in the negative direction of both capacitors. The store phase integrates

on C_{int2} and the subtraction phase uses C_{int1} . The state of switches $sw1$ - $sw6$ is the complement of the odd frames. At the end of the integration time, the CTIA outputs the voltage across capacitor C_{int2} for odd frames and C_{int1} for even frames, which represents the frame-difference value. The voltage across the capacitors at the end of the integration time is shown in Eq. (4.7):

$$V_c = (I_k \Delta t_s - I_{k-1} \Delta t_s) / (2C_{int}), \quad (4.7)$$

where V_c is the output voltage, k is the current frame index, I_k is the input current during frame k , I_{k-1} is the current during the previous frame $k - 1$, Δt_s is the integration time where $\Delta t_s = \frac{\Delta t}{2}$, and C_{int} is the capacitance of C_{int1} and C_{int2} . After the frame-difference values are read by the A-THR core, the circuit resets the capacitor that holds the frame-difference value.

Note that, because the circuit store and subtract in different directions during odd and even frames, the frame-difference in V_c has different sign in consecutive frames. This sign difference does not affect the results of the algorithm because the next stages use the absolute value of the difference. Alternating the sign of the frame-differences allows us to configure the CTIA using only three control signals. Moreover, $sw1$ - $sw6$ switch only once per frame instead of once per phase, which reduces charge injection and power consumption.

When the smart pixel operates in frame-difference mode, its integration time is the 50% of the time that is used to operate in conventional mode. This reduction decreases the signal-to-noise ratio of the pixel. However, the smart pixel can compute the frame-differences at the same time on the entire array of smart pixels with small impact on the pixel fill factor.

4.2.2 A-THR

The A-THR core determines whether the absolute value of the frame-difference computed by the smart pixel exceeds an application-defined threshold. Figure 4.16 shows the A-THR circuit. A row- and column-select circuit scans the smart pixel array, reading the output voltage of each CTIA connecting it to the input V_{pixel} of the A-THR module. Because the frame-difference output of the smart pixel has a different sign for even and odd frames, the A-THR core uses two comparators OA1 and OA2. The reference voltages V_{THR}^+ and V_{THR}^- are used to compare the absolute value of the frame-difference voltage to the threshold voltage V_{th} , such that $V_{THR}^+ = V_{bias} + V_{th}$ and $V_{THR}^- = V_{bias} - V_{th}$, where $V_{bias} = V_{dd}/2$ is the bias voltage of the CTIA in frame-difference mode.

The comparator OA1 outputs a logical 1 when $V_{pixel} > V_{th}^+$, and 0 otherwise, while OA2

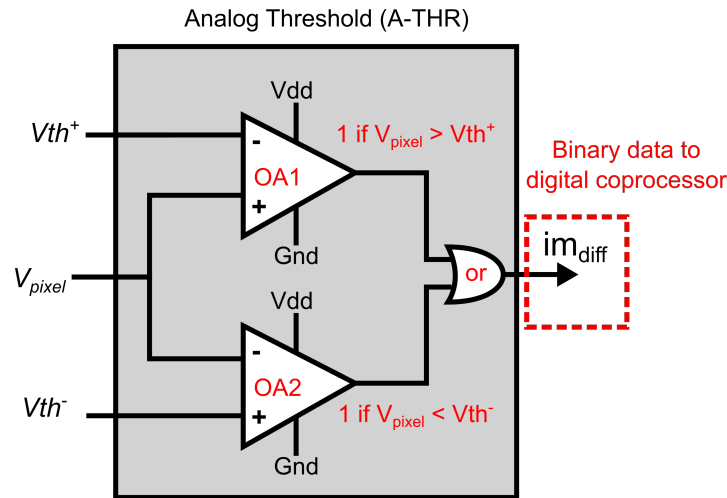


Fig. 4.16: Architecture of the A-THR. An input comparator compares the frame-difference for each pixel to two reference voltages. The comparator OA1 outputs a logical 1 when $V_{pixel} > V_{th}^+$, and the comparator OA2 outputs a logical 1 when $V_{pixel} < V_{th}^-$. A logical OR outputs a logical 1 if one of the two conditions is met.

outputs a logical 1 when $V_{pixel} < V_{th}^-$, and 0 otherwise. These two comparators independently indicate when V_{pixel} is greater than V_{th}^+ or less than V_{th}^- . An OR gate outputs a logical 1 when either OA1 or OA2 output a 1, thus indicating that the frame-difference is greater than the supplied threshold.

4.2.3 Digital coprocessor

The coprocessor adds programmability to the SIS by processing the output in frame-difference mode using reconfigurable digital logic. In the current implementation, the coprocessor implements the morphological opening operation and a connected components algorithm that detects objects and computes their bounding boxes.

Figure 4.17 shows the architecture of the object detection coprocessor. The data flow of the digital coprocessor is as follows: the object detection coprocessor receives a 1-bit pixel stream from the A-THR module. Then, the coprocessor computes a morphological erosion and dilation operations on 3×3 -pixel window and outputs the resulting binary image. The image pixels are also processed by a connected components module, which identify the objects in the image using connected pixels in a single pass and computes the bounding boxes of the objects.

The digital implementation of the 1-bit image erosion, defined in Eq (3.2), is shown in Figure 4.18. The circuit erodes with a 3×3 window, by calculating the logical AND between all pixels

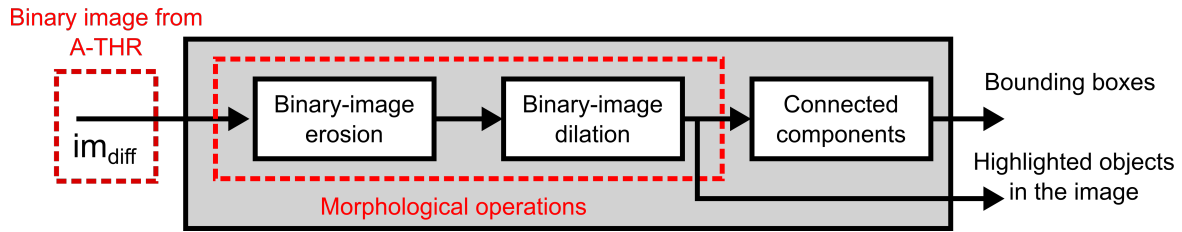


Fig. 4.17: Architecture of the digital coprocessor. The coprocessor receives a stream of movement pixels, applies morphological opening operation (erosion+dilation), and computes the connected components of the resulting binary image and their bounding boxes.

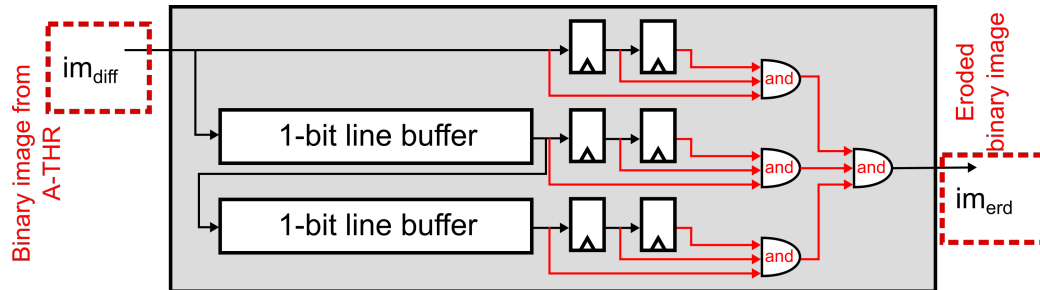


Fig. 4.18: Image erosion. The module uses two line buffers and six registers to define a 3×3 -pixel window from the output of the smart pixel array, and performs image erosion by computing a logical AND operation between them.

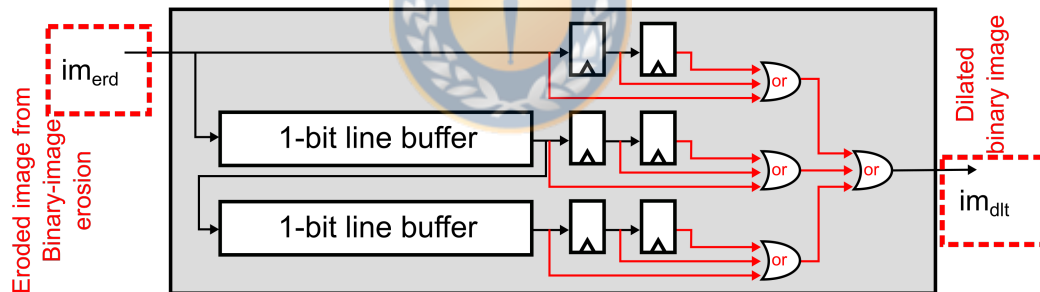


Fig. 4.19: Image dilation. The module uses two line buffers and six registers to define a 3×3 -pixel window from the output of the image erosion module, and performs image dilation by computing a logical OR operation between the pixels.

in the window. The sliding window was implemented using two line buffers and a 2×3 array of flip-flops. Figure 4.19 shows the implementation of the 1-bit image dilation, defined in Eq (3.3). Dilation's methodology is similar to the erosion, which uses the same architecture but replaces the logical operations with OR gates.

Figure 4.20 shows the architecture of the connected components module. The Neighborhood Context block uses a line buffer to define a 2×2 -pixel window that contains the current pixel and its north, northwest and west neighbors. The block output indicates whether the current

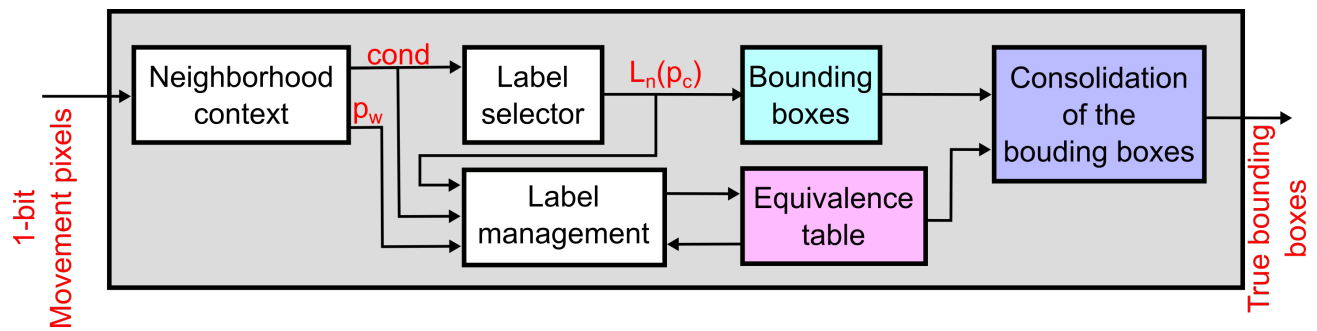


Fig. 4.20: Architecture of the connected components module. First, it analyzes the current pixel and its north, northwest and west neighbors, determining which movement pixels are connected. The module assigns a label to the current pixel and maintains an equivalence table to merge connected components in the image. The module also computes the bounding boxes for all connected components and merges them using the equivalence table.

pixel is an isolated movement pixel, to which of its neighbors it is connected, or whether it is not a movement pixel. The Label Selector block assigns a new or existing label to the current pixel based on its neighboring labels, using a line buffer with label information. Because new pixels can join disconnected regions, the module uses an equivalence table to merge connected components. The Label Management block updates the equivalence table using the information from the neighborhood context. As new pixels are added to the existing connected components, the coordinates of their bounding boxes are updated using the contents of the equivalence table to consolidate regions as they merge.

5. Results

5.1 Evaluation Methodology

The following methodology was used to evaluate and obtain the results of the designed SISs. The SISs evaluation was divided into two steps, smart-pixel array simulations and digital coprocessor implementation and validation.

The smart-pixel array simulations included the following. Spice simulations validate the smart-pixel architecture's functional operation, considering ideal components with their respective transistor size. Physical-layout design to extract the parasitic components of the circuit and determine the readout-circuit area. Use a Layout vs. Schematic (LVS) evaluation tool to validate the physical-layout netlist. Post-layout simulation to validate that the smart pixel circuit is still functioning as desired. The circuitry column level also passed the process described from the beginning. The final evaluation of the smart-pixel array included the simulation of the complete array interconnected and the circuitry at a column level. The post-layout results were written into a collection of time-voltage values representing all the important signals. This collection was then used as input to the digital coprocessor emulation. An important metric to evaluate and compare the smart pixel is the fill factor. The pixel's fill factor FF_{pix} is defined as the ratio between the and the photosensitive area and the total pixel area such that $FF_{pix} = 1 - \frac{A_r}{A_p}$, where A_r is readout-circuit area and A_p is the pixel area.

The implementation results of the digital coprocessor were obtained by writing an Hardware Description Language (HDL) code that described the digital architecture at the register Register-Transfer Level (RTL). To evaluate the digital coprocessor functionality, the RTL description was synthesized and implemented using Vivado's synthesis tool. The resulting bit-stream was programmed into an Field Programmable Gate Array (FPGA) to test different input combinations and evaluate its results at the output. The input and output data were transferred to a desktop computer using an external memory card. Then, its results were compared to a fully-software implementation. The numerical results of the digital coprocessor were obtained by writing a software script that emulated each step of the hardware implementation, that is, a fixed-point representation of the results in all the stages of the pipeline.

Finally, to fully evaluate the SIS results, the post-layout simulations of the smart-pixel

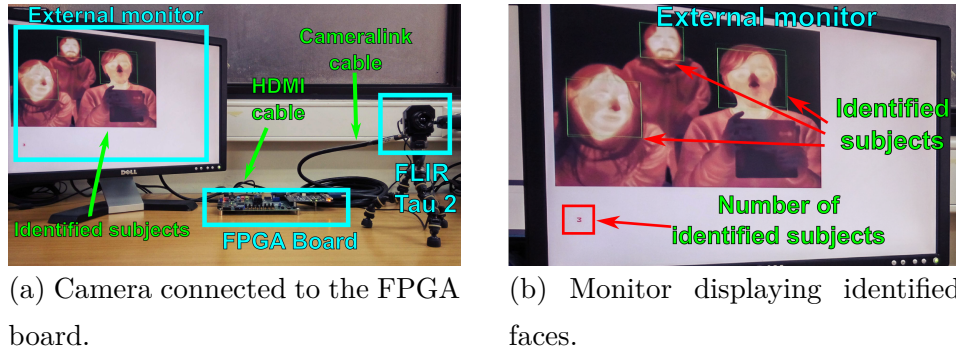


Fig. 5.1: Experimental setup to test the face-recognition algorithm. An FPGA board receives IR images from a FLIR Tau 2 camera core and uses a HOG algorithm to detect face locations. The FPGA emulates the smart pixel array and the digital coprocessor. A monitor connected to the FPGA displays the image acquired by the smart pixel array, and the location and number of identified faces. The FPGA sends the labels of the recognized faces to a remote computer via Ethernet.

array were used as the input of the digital-coprocessor fixed-point emulation. The input of the smart-pixel array are currents values that are proportional to pixel values of an input image within the range of $[0, 255]$ (8-bit representation), where the minimum pixel-value is converted as $0 = 0A$, and the maximum as $255 = I_{max}$ (I_{max} defined by the smart-pixel design). The resulting voltage values of the pixel-array simulation were digitalized and used as the input of the digital-coprocessor emulator.

5.2 Face recognition results

Figure 5.1 shows an experimental setup used to validate the proposed face recognition method using an FPGA connected to a thermal IR FLIR Tau 2 camera core. In this case, the FPGA runs a face detection algorithm to locate faces in the acquired image, emulates the smart pixel array that computes the local spatial gradients on the image and the RPG that generates the URLBP values for each pixel, and implements the digital coprocessor. The FPGA outputs the acquired image on an external monitor, and sends out the labels of the recognized faces via an Ethernet link.

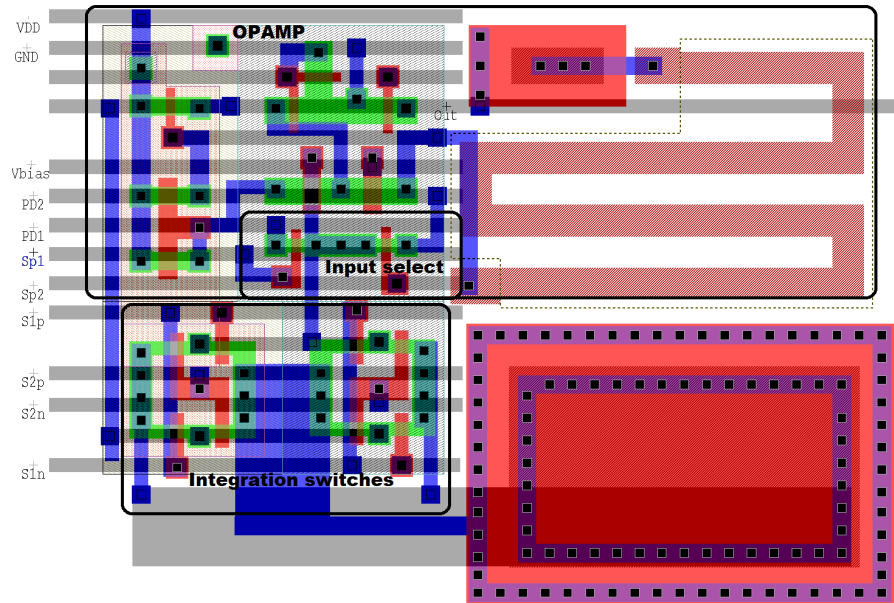


Fig. 5.2: Layout of the smart-pixel. This work used the design shown in Fig. 4.3, implemented on the TSMC 0.35 μm mixed-signal process. The opamp and integration capacitors are implemented using two poly layers.

5.2.1 Smart pixel and RPG implementation

Figure 5.2 shows the layout of the smart-pixel circuit depicted in Fig. 4.3 using the TSMC 0.35 μm mixed-signal process with a 3.3 V supply voltage. A poly1-poly2 capacitor was used for integration, which has a capacitance per area of $950 \text{ aF}/\mu\text{m}^2$. Assuming an integration time of $40 \mu\text{s}$ and a maximum photodetector current of 8 nA , the pixel requires a 100 fF integration capacitor of $13.6 \mu\text{m} \times 7.7 \mu\text{m}$. The dimensions of the complete circuit, including all passive and active elements, are $30 \mu\text{m} \times 22.5 \mu\text{m}$. Assuming a standard $32 \mu\text{m} \times 32 \mu\text{m}$ pixel [41], the circuit achieves a fill factor of 34%. The extra transistors used to compute local gradients increase the area of the circuit by 26%. Without the switches used to operate in smart mode, the fill factor is 47.6%.

The design of the smart pixel was ported to the 0.18 μm TSMC process, which is more commonly used in the literature [41, 42, 44]. With a 1.8 V supply voltage and $2 \text{ fF}/\mu\text{m}^2$ metal capacitors, the total area of the circuit is $243 \mu\text{m}^2$, which allows us to achieve a fill factor of 76% in the same $32 \mu\text{m} \times 32 \mu\text{m}$ pixel. In comparison, the integration circuit without the switches for local gradient computation has a fill factor of 79.9%. In summary, the presented smart pixel is capable of computing spatial differences during integration with a small impact on the fill factor.

Table 5.1: Comparison of the proposed smart-pixel designs to other circuits in the literature.

SIS	Technology	Pixel pitch $\mu m \times \mu m$	Fill factor	Tested spectrum	Type of integrator
Proposed RLBP+LDA face recognition	0.35 μm TMSC	32×32	34%	Visible Thermal IR NIR	CTIA
	0.18 μm TMSC	32×32	76%	Visible Thermal IR NIR	CTIA
Edge detection [41]	0.18 μm 1P4M CMOS	31×31	19%	Visible	2T-integrator, CTIA at column level
LBP Edge detection [42]	0.18 μm CMOS	7.9× 7.9	55%	Visible	2T-integrator
Spatial contrast LBP [129]	0.35 μm	26 × 26	23%	Visible	2T-integrator
4-neighbor LBP [71]	0.35 μm CMOS	64×64	15%	Visible	2T-integrator

Table 5.1 compares the smart pixel to other designs reported in the literature and discussed in Section 2. Even though the smart pixel described in [41], which computes local differences for edge detection, uses a CTIA only at the column level, it reaches a fill factor of 19%, which is much lower than the 76% reached by the proposed solution in a similar CMOS process. The other designs shown in the table use a much simpler 2-transistor integrator, which is only suitable for capturing images in the visible spectrum. Nevertheless, the proposed design achieves a better fill factor in all cases.

Figure 5.3 shows a post-layout simulation of the CTIA during the positive and negative integration phases. The figure depicts the capacitor voltage V_{Cint} for five different pixels in the FPGA. During the positive integration phase, the output voltages increase proportionally to the local photodetector current of the pixel. Then, during the negative phase, the voltage decreases at a rate proportional to the photodetector current of the horizontally adjacent pixel. The voltage after the negative phase is proportional to the difference between the two pixel values. The output voltage $V_{Cint} + V_{bias}$ is then sampled when reading the pixel value.

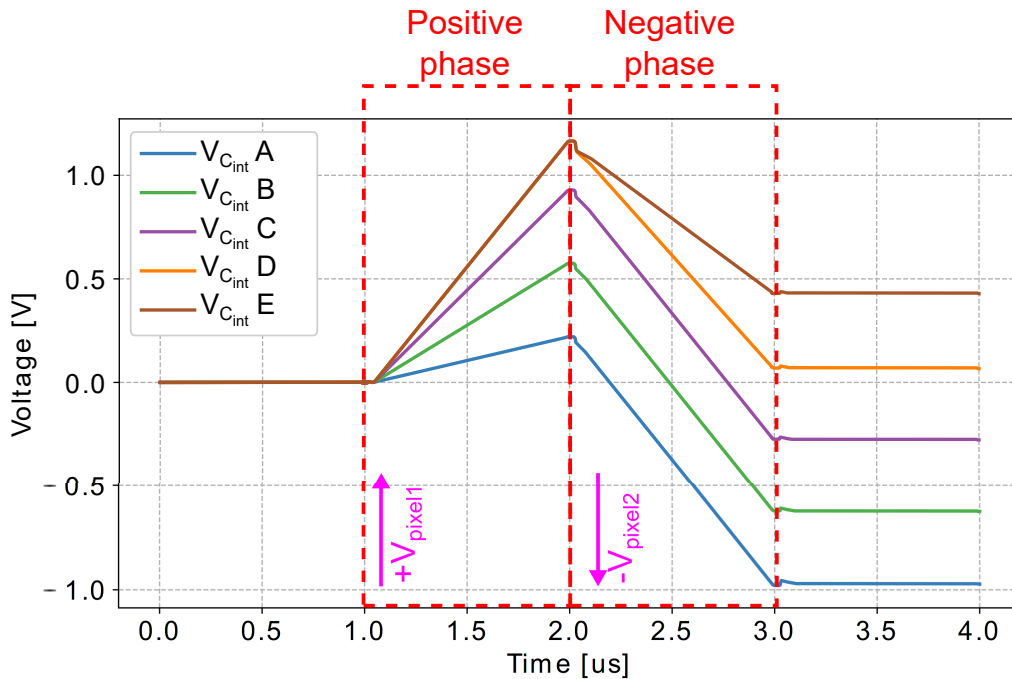


Fig. 5.3: Post-layout simulation of five pixels in the SIS operating in local-gradient mode. The graph shows the voltage across the integration capacitor of the CTIA in Fig. 4.14 during the positive and negative integration phases shown in Fig. 4.5.

The simulation plot in Fig. 5.4 shows the operation of the input comparator of the RPG in Fig. 4.6. During the integration phase, all the pixels compute their local horizontal gradient in parallel. For clarity, Fig. 5.4 shows the output voltage of two pixels (Pixel A and Pixel B). During the comparison phase, the controller performs a row-wise read of the pixel array, sequentially reading the values of three vertically-adjacent cells for each pixel, as described in Section 4.2.2. The voltage labeled V_{column} shows the array output voltage, which is the input to the comparator. The output values of Pixel A and Pixel B are the first and third voltages presented to the comparator, respectively, and are sampled at the times circled in red in Fig. 5.4. When the input voltage is higher than V_{ref} , the comparator outputs V_{dd} , or a logic 1. Otherwise, it outputs 0V, or a logic 0. In the simulation shown in the figure, the comparator outputs the logic sequence 00110001011, which is delivered to the shift registers of the RPG generator to create the RLBP values.

When simulating the operation of an array of 150×80 pixels in the $0.35 \mu\text{m}$ process, including post-layout parasitics, the readout time of one pixel is 50ns. Considering that the circuit read each pixel 3 times in smart mode, the readout time for the complete array is 1.8ms, which allows us acquire and process images at 556 fps.

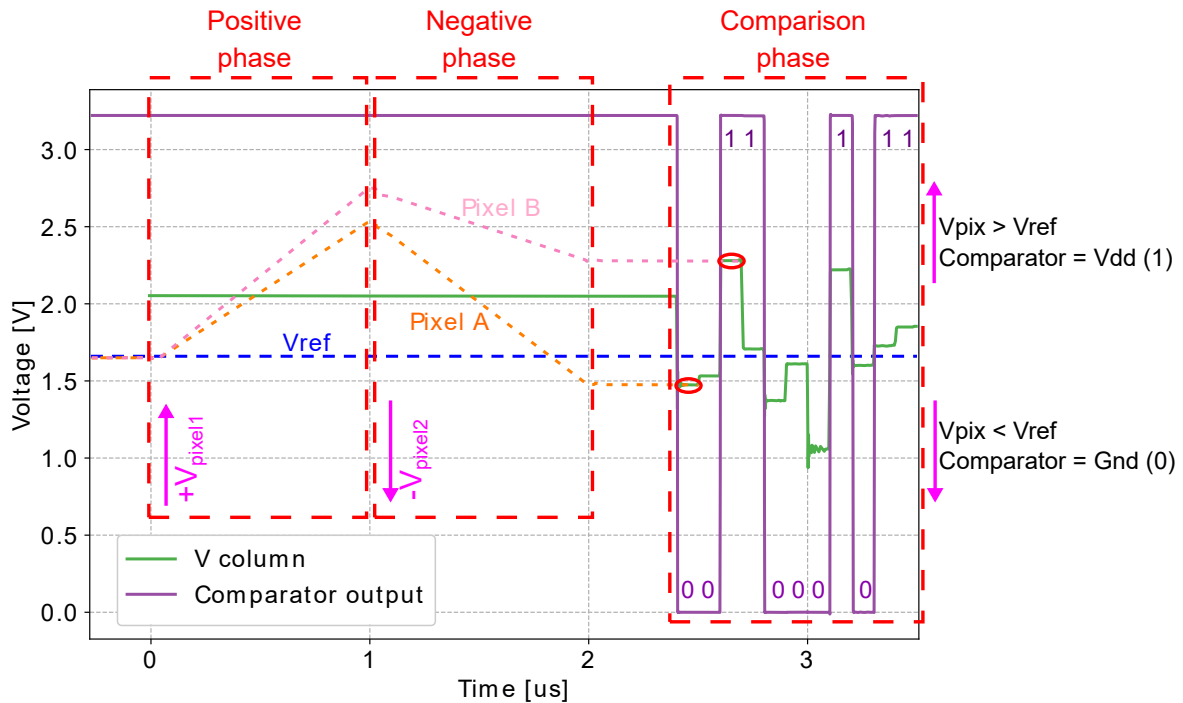


Fig. 5.4: Post-layout simulation of the RPG input comparator while reading multiple pixels. The plot shows the integration phase for the first and third pixel, the voltage input to the comparator, the reference voltage, and the voltage output. Gradient values are read every 50ns.

Table 5.2: Resource utilization of the digital coprocessor on a Xilinx XC7Z020 FPGA.

	Slice LUTs	Distributed memory	Block RAMs	DSP slices
Total used	4345	1298	2	6
Available	53200	17400	140	220
Percentage	8.2%	7.6%	1.4%	2.7%

5.2.2 FPGA implementation of the digital coprocessor

The architecture of the digital coprocessor was modeled at the register-transfer level using the System Verilog hardware design language, and synthesized the design onto a Xilinx XC7Z020 FPGA. Table 5.2 shows the resource utilization of the proposed design. The coprocessor uses less than 10% of the logic and distributed memory available on the chip. Distributed memory was used to implement the coefficient buffers in Fig. 4.8, because the buffers are small (64 entries), and they need to be accessed in parallel to obtain a new set of coefficients with each uniform RLBP element. The buffer in the Euclidian distance module of Fig. 4.9 uses two embedded RAM blocks, which account for 1.4% of the blocks available on the chip. Finally, the same module uses six out of the 220 DSP slices available on the FPGA. This small hardware resource usage leaves ample space on the FPGA to implement additional image processing algorithms.

Table 5.3: Databases used to test the performance of the proposed method.

Database	Spectrum	Image size (pixels)	Number of subjects	Images per subject	Face positions and conditions
UCH-TF	Thermal IR	150×81	53	28	Rotations and expressions
CBSR NIR	Near IR	640×511	197	20	Frontal, with and without glasses
UL-FMTV	Thermal IR	128×128	238	Short video sequence	Rotations
YaleFace B	Visible range	192×168	38	64	Frontal, expressions and light variations

The coprocessor operates at a maximum clock speed of 128MHz, mostly due to the need to access external memory for the LDA coefficients and the database of stored faces. At this clock rate, the circuit can process 127×10^6 pixels per second or one 150×80 -pixel image in $94 \mu\text{s}$. The power consumption of the circuit operating at this clock frequency, estimated using Xilinx Power Analyzer, is 71mW.



5.2.3 Method classification performance

To test the classification performance of the proposed method, four databases were used: UCHThermalFace database [130], the CBSR NIR face dataset [131], the Université Laval Face Motion and Time-Lapse Video Database (UL-FMTV) [132, 133], and the Yale Face Database B [134]. Table 5.3 summarizes the information of spectrum, image size in pixels, number of subjects, number of images per subject, variations in face position and other conditions of the images in each database.

To evaluate the classification performance of the algorithm on each database, 60% of the images were used for training, that is, to compute the LDA transformation and the stored database of projected faces, and 40% for testing. To reduce overfitting, a standard k-fold cross-validation technique was used with 10 iterations. The performance of the algorithm was quantified using the accuracy score, which is defined in Eq. (5.1) for a multiclass classification problem with N classes:

$$\text{Accuracy} = \frac{\sum_{i=1}^N H_i}{\sum_{i=1}^N H_i + F_i} \times 100, \quad (5.1)$$

Table 5.4: Accuracy of the proposed method with RLBP and LBP using different databases.

	UCH-TF	CBSR NIR	UL-FMTV	YaleFaceB
RLBP+LDA	96.7%	96.0%	75 – 95.9%	76.4%
LBP+LDA	98.5%	98.1%	79 – 97.1%	82.9%

Table 5.5: Accuracy of the proposed RLBP+LDA and LBP+LDA methods compared to other face classification algorithms discussed in [75], using the UCH-Thermafalce [130] database.

Method	RLBP+LDA	LBP+LDA	GJD	WLD	LBP
Accuracy	96.7%	98.5%	96.6%	94.4%	92.0%

where H_i and F_i are the number of correctly and incorrectly labeled samples of class i , respectively. In other words, the accuracy is the percentage of correctly classified images in the test set, computed as the sum of the diagonal elements of the confusion matrix divided by the sum of all the elements in the matrix.

Table 5.4 shows the accuracy of the proposed method on the four databases, using both RLBP and conventional LBNP to compute the local features. As the table shows, the proposed method performs best with the UCH-TF and CBSR NIR databases, which consist mainly of frontal images with small variations in rotation. The achieved accuracy is lower, but still above 80%, for YaleFaceB, which is a challenging dataset with significant variations in illumination among images. The UL-FMTV contains short video sequences, of which 24 images were extracted for training and 16 for testing, for each subject. The classification accuracy depends largely on which video frames were used to train and test the algorithm: The accuracy is lowest, but still around 75%, when the training and testing frames are far apart in the video frame, mostly due to large variations in rotation angle. When the sets are taken from closer video frames, the algorithm achieves accuracy above 95%.

Table 5.4 also shows that replacing conventional LBP with the proposed lightweight RLBP descriptor reduces classification accuracy in approximately 2 – 5%, depending on the database. LBP considers gradients in all directions in a 3×3 -pixel window, while RLBP groups only horizontal gradients in the same window. Although LBP contains more information than RLBP, RLBP can still capture significant texture information because it considers the spatial distribution of horizontal gradients within a small pixel neighborhood while reducing the number of operations at the pixel level.

Tables 5.5, 5.6 and 5.7 compares the accuracy of the proposed method to other algorithms in the literature using the databases with which they were published. Table 5.5 reports the

Table 5.6: Accuracy of the proposed RLBP+LDA and LBP+LDA methods compared to other face classification algorithms discussed in [74], using the CBSR NIR [131] database.

Method	RLBP+LDA	LBP+LDA	NIRFaceNet+Aug	NIRFaceNet	FaceNet
Accuracy	96.0%	98.2%	96.6%	94.8%	84.1%

Table 5.7: Accuracy of the proposed RLBP+LDA and LBP+LDA methods compared to other face classification algorithms discussed in [135], using the Yale face database B [134].

Method	RLBP+LDA*	LBP+LDA*	Sun’s Kernel	CRC	ELM	Tanh
Accuracy	76.4%	82.9%	98.33%	96.82%	96.44%	96.34%

algorithms evaluated by Hermosilla et. al [75] using the UCHThermalFace database. Table 5.6 shows the accuracy achieved with the algorithms reported by Jo et. al [74] with the CBSR NIR database. In all these cases, the method achieves similar or better accuracy than the algorithms reported in the literature. Finally, Table 5.7 compares the results of the proposed method with YaleFace B database against the algorithms evaluated by Sun et. al [135]. The results show that the accuracy of the proposed algorithm is lower than the reported methods. The main reason for this is that Ahonen’s algorithm does not perform well when there are large variations in illumination between the images in the training and test set. Sun’s algorithm shows more robustness under these conditions, but it requires more computation per pixel. Moreover, this computation can not be easily mapped onto a smart pixel design in the analog domain to exploit pixel-level parallelism in the imager. For NIR and thermal IR images, for which the proposed smart pixel is better tailored, the proposed method delivers better results than the state of the art.

All the experiments described above were executed as a closed-set problem, that is, the test set contains only images of subjects that are also present in the training set. In order to test the performance of the proposed method in an open-set problem, the classifier was trained using only 40 subjects from the UCHThermalFace database, and used a test set with images from all 53 subjects. The threshold THR described in Algorithm 1 was used to label the image as unknown if the distance to its nearest is larger than this threshold. In this experiment, using $THR = 8$, the accuracy of LBP+LDA is 95.5%, which is reduced to 93.1% when using RLBP to compute the local features. That is, the accuracy of both methods is reduced by a approximately 3% compared to the closed-set problem.

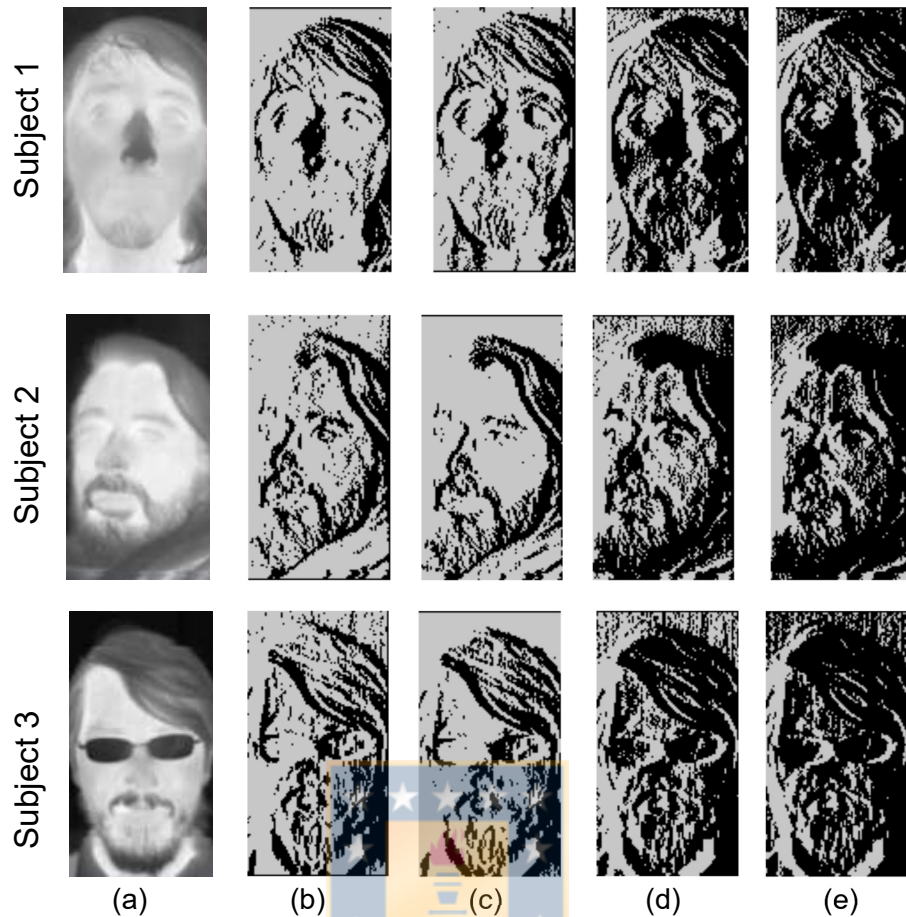


Fig. 5.5: Effect of V_{ref} in the RLBP values generated by the RPG for 3 images acquired using a FLIR Tau 2 thermal IR core. (a) original IR image, (b) RLBP image generated by software, (c)-(e) RLBP images generated by the RPG for V_{ref} values of 1.665 V, 1.650 V and 1.645 V, respectively.

5.2.4 SIS classification performance

A circuit parameter that affects classification performance is the reference voltage in the comparator of Fig. 4.6, which computes a digital value for the difference between adjacent pixels. The images in Fig. 5.5 illustrate the effects of V_{ref} in the RLBP values generated by the comparator, for thermal IR images of 3 different subjects. Fig. 5.5 (a) shows the original image, Fig. 5.5 (b) is the image generated by replacing the pixel values with the RLBP values computed in software, and Fig. 5.5 (c)-(e) are the RLBP images generated by the hardware setting the value of V_{ref} to 1.665V, 1.650V, and 1.645V.

Figure 5.6 shows the accuracy achieved in the simulations by the complete circuit as a function of V_{ref} using the UCHThermalFace database [130]. For comparison, the figure also shows the classification accuracy achieved by a software implementation of the algorithm, and

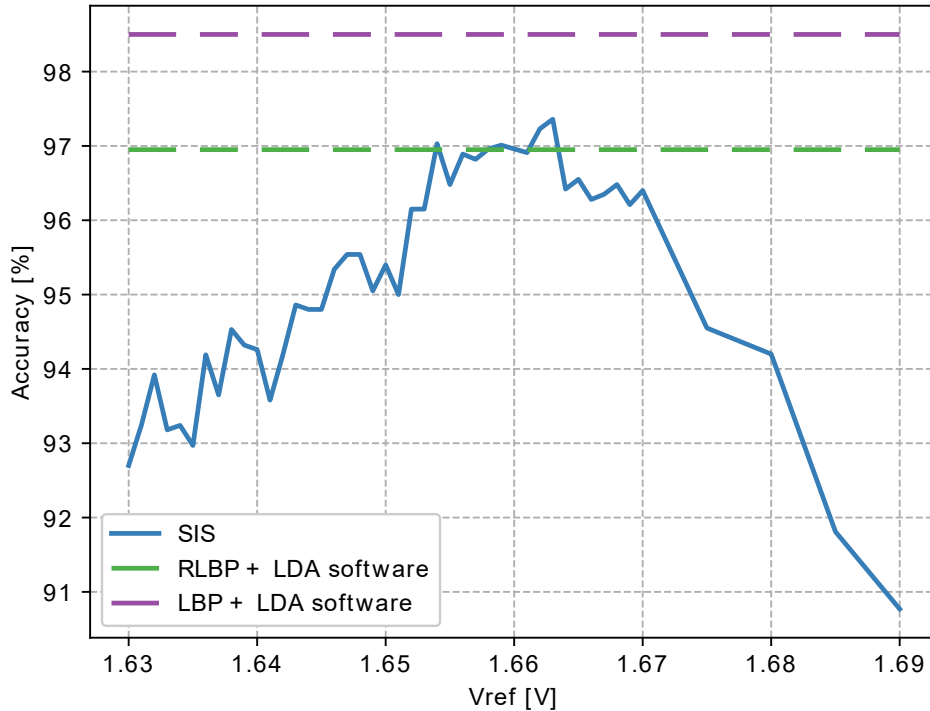


Fig. 5.6: Classification accuracy as a function of the value of the comparator input V_{ref} in Fig. 4.6.

by the same algorithm in software, using conventional LBP instead of the proposed RLBP. When programmed in software, the proposed algorithm achieves 96.7% accuracy on the test dataset, while using LBP achieves 98.5%, but requires 8 times as many comparisons. Varying V_{ref} between 1.63 V and 1.68 V, the proposed hardware implementation using the SIS and digital coprocessor on the FPGA achieves an accuracy above 93%. Setting V_{ref} between 1.655 V and 1.665 V achieves a mean accuracy of 96.5%. These values are slightly higher than the expected value of $V_{ref} = V_{dd}/2 = 1.65V$, mostly because of change injection in the feedback and parasitic capacitors of the comparator.

5.3 Object detection results

5.3.1 Smart pixel and A-THR implementation

Figure 5.7 shows the physical layout for the smart-pixel presented in Fig. 4.3 to detect objects. This design uses a $0.35 \mu\text{m}$ mixed-signal process, $950 \text{ aF}/\mu\text{m}^2$ poly1-poly2 capacitors, and a supply voltage of 3.3 V. The integration time of the proposed smart pixel is $20 \mu\text{s}$ and the maximum current that the photodetector delivers is 8nA . With this, the two integration

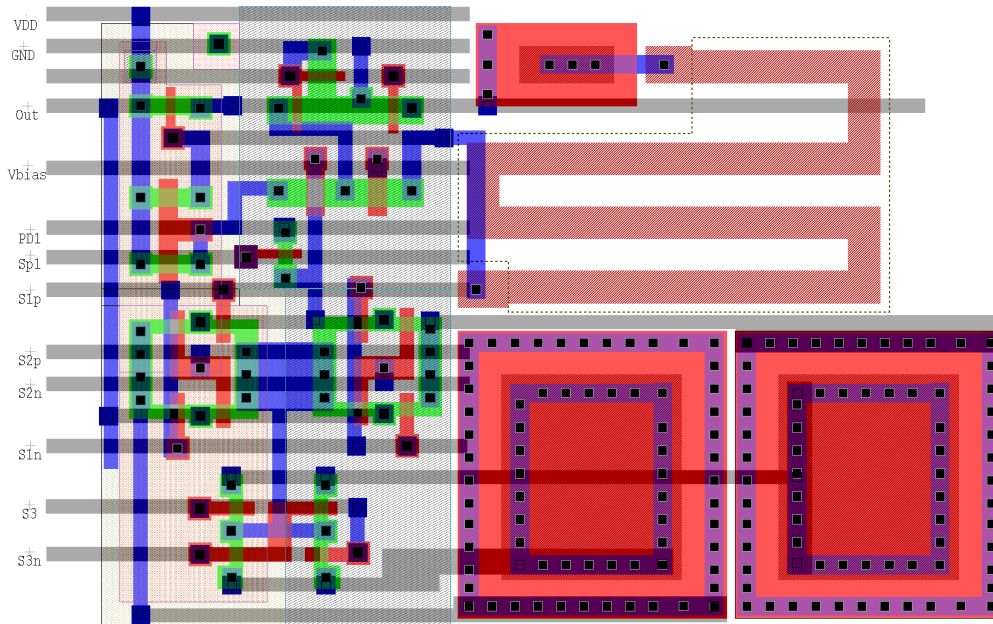


Fig. 5.7: Diagram of the smart-pixel layout. Using the design shown in Fig. 4.3 and implemented on the TMSMC 0.35 μm mixed-signal process. The opamp and integration capacitors are implemented using two poly layers.

capacitors have an equal capacitance of 50 fF with a size of $7.7 \mu\text{m} \times 7.7 \mu\text{m}$. The area of the entire smart-pixel circuit is $32 \mu\text{m} \times 23 \mu\text{m}$, which achieves a fill factor of 28% in a standard $32 \mu\text{m} \times 32 \mu\text{m}$ pixel [65]. In comparison, a conventional CTIA circuit designed on the same process has a fill factor of 47.6%.

The layout of the smart pixel was drawn using the 0.18 μm TMSMC process, a technology commonly used in the literature [44, 65, 70]. For this technology a supply voltage of 1.8 V and metal capacitors of $2 \text{ fF}/\mu\text{m}^2$ capacitance were used. The size of the circuit is $14 \mu\text{m} \times 19 \mu\text{m}$, which achieves a fill factor of 74% in the same $32 \mu\text{m} \times 32 \mu\text{m}$ pixel, compared to 86.3% with the conventional CTIA.

Figure 5.8 shows a post-layout simulation of a CTIA pixel of the SIS operating in frame-difference mode. The figure depicts the voltage across the capacitors C_{int1} and C_{int2} during two consecutive video frames. In the odd frames, during the store phase the capacitor voltage $V_{C_{int1}}$ starts at zero and increases linearly with the photodetector current, and the voltage $V_{C_{int2}}$ represents the pixel value of the previous frame. During the subtract phase, $V_{C_{int1}}$ stays constant and $V_{C_{int2}}$ decreases linearly with the photodetector current. After the subtract phase, $V_{C_{int2}}$ is proportional to the differences between the pixels in the two consecutive frames. The output voltage $V_{C_{int2}} + V_{bias}$ is then sampled when reading the pixel value. For the even frames, the

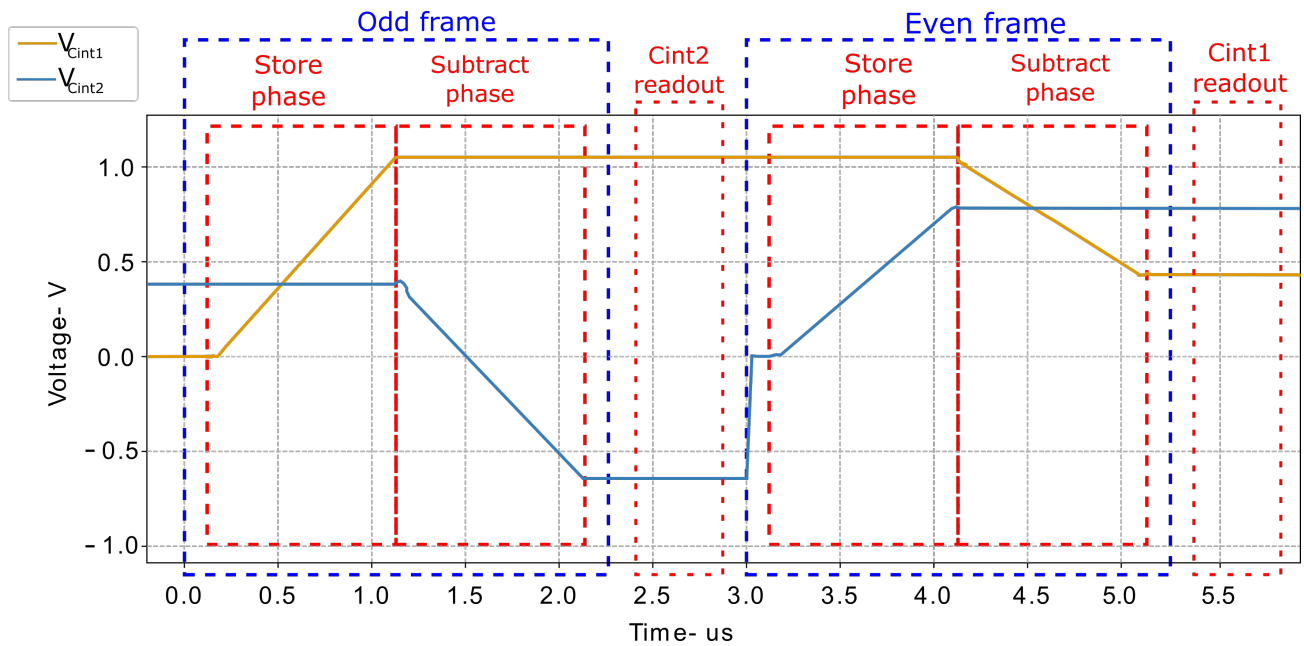


Fig. 5.8: Post-layout simulation of a pixel in the SIS operating in frame-difference mode. The graph shows the voltage across the two integration capacitors of the CTIA during two consecutive frames.

role of the capacitors is reversed, and the circuit outputs $V_{Cint1} + V_{bias}$ to the next stage.

The simulation plot in Fig. 5.9 shows the operation of the A-THR comparator of Fig. 4.16. During the subtract phase, all pixels compute their respective frame-difference. For clarity, Fig. 5.9 shows the output voltage of two pixels (pixels A and B). During the readout and comparison phase, the controller performs a column-wise read of the pixel array. The readout voltage of a column is labeled V_{column} . The value of two different pixels, A and B, in the same column inputs the A-THR. Those value, and the rest of the pixel values on the same column, are sampled during the times circled in purple. If the input voltage is outside the threshold window, i.e., greater than V_{THR}^+ or less than V_{THR}^- , the A-THR outputs a logic 1, and outputs a logic 0 otherwise. The simulation in Fig. 5.9 shows that the output of the A-THR is a logic sequence 11001011100. This sequence is then delivered to the digital coprocessor.

5.3.2 FPGA implementation of the digital coprocessor

The SystemVerilog HDL at the RTL was used to implement and validate the architecture of the digital coprocessor using the Xilinx Vivado 2020.1 development platform. In order to showcase the reduction in digital hardware resources enabled by the SIS, the low-cost entry tier Xilinx Artix-7 XC7A35T FPGA was targeted. It was compared the results to an FPGA-based Fully-

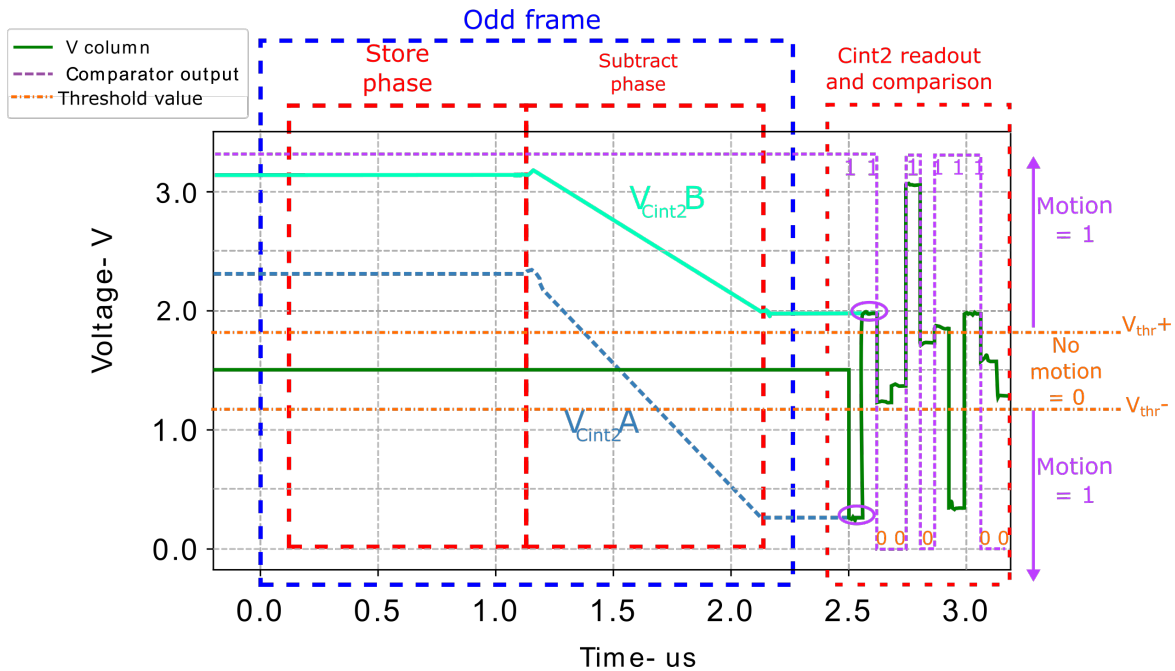


Fig. 5.9: Post-layout simulation of the A-THR comparator while reading multiple pixels in frame-difference mode. The plot shows the subtract phase for two pixels A and B in the same column, during an odd frame. In the readout phase, the comparator consecutively samples all pixels in each column, comparing their value to the application-defined thresholds and output a logic 1 when the movement in the pixel exceeds the threshold. Pixel values are sampled every 50 ns.

digital implementation (FDI) of the algorithm that uses a conventional image sensor. The FDI operates on 8-bit grayscale pixels. All implementations use 5-bit labels and a 32-entry equivalence table in the Connected components module. Two tests scenarios with different input image resolutions were considered: 320×240 and 640×480 pixels.

Table 5.8 shows the resource utilization of both implementations for both image resolutions. The proposed coprocessor architecture using the SIS requires 5930 and 3929 Lookup tables (LUTs) for the 640×480 and 320×240 -pixel implementations, respectively. This represents 28.5% and 18.8% of the LUTs available on the XC7A35T FPGA. The presented implementations

Table 5.8: Resource utilization of the digital coprocessor on a Xilinx Artix-7 XC7A35T FPGA.

	SIS 640x480		FDI 640x480		SIS 320x240		FDI 320x240	
	Util	%	Util	%	Util	%	Util	%
LUT	5930	28.5	6493	31.2	3929	18.8	4051	19.4
FF	5021	12.0	5107	12.2	3239	7.7	3270	7.8
BRAM	0	0	75	150	0	0	19	38

also utilize 12% and 7.7% of the available Flip-Flop (FFs). No on-chip BRAM are required in the SIS-based architecture. When compared with the SIS-approach, the FDI needs a frame buffer to compute the temporal differences between pixels in consecutive frames, which is implemented with Block Random Access Memory (BRAM) to avoid using an external memory chip, which would limit the performance of the algorithm and increase the overall cost of the system. Indeed, the 320×240 -pixel FDI requires only a small increase in the utilization of LUTs and FFs, but uses 38% of the available BRAM. Moreover, the 640×480 -pixel FDI requires more BRAM resources than those available on the FPGA, and thus could not be implemented on the selected device. The small hardware utilization of the proposed SIS-based coprocessor leaves ample resources available, even on an entry-level device such as the XC7A35T FPGA. These resources could be used to implement additional image-processing algorithm on the output produced by the SIS.

Table 5.9 shows the power consumption of the coprocessor estimated by Xilinx Vivado. Operating with the 20 MHz clock frequency imposed by the sampling rate of the SIS, the power consumption of the presented coprocessor is 27 mW and 34 mW for the 320×240 and 640×480 -pixel inputs, respectively. The coprocessor can operate at up to 125 MHz, which enables it to operate at up to 1627 fps on 320×240 -pixel images while consuming 58 mW, and at up to 406 fps on 640×480 -pixel images while consuming 61 mW. In comparison, the FDI with 320×240 -pixel input consumes 39 mW at 20 MHz, and 97 mW at its maximum clock frequency of 104 MHz. Here, the power consumption of the frame buffer, implemented as an on-chip memory, is near the 50% of the total dynamic power. At this frequency, the FDI can operate at up to 1354 fps.

5.3.3 SIS object detection performance

To test the performance of the motion-based object-detection algorithm on the proposed SIS, the OSU Thermal Pedestrian [136] and the Terravic Motion Infrared (IR) [137] databases were used. Both contain video sequences in the thermal IR range. Table 5.10 summarizes the image size in pixels, number of video sequences and total number of images.

To evaluate the object location performance of the SIS on each dataset, we used a simulation of the complete SIS circuit with post-parasitic extraction and the FPGA-based coprocessor described in Section 5.3.1. We developed a software implementation of the algorithm using floating-point arithmetic and used it as a baseline to evaluate the performance of the algorithm on the SIS.

Table 5.9: Power consumption of the digital coprocessor on a Xilinx Artix-7 XC7A35T FPGA, estimated by Vivado. All implementations consume 20 mW of static power, which are added to the dynamic power to compute the total. The 640×480 -pixel FDI can not be implemented on the XC7A35T device.

	Dynamic power (mW)				Total Dynamic (mW)	Total (mW)
	Dilation	Erosion	Connected components	Frame Buffer		
SIS 320x240 (20MHz)	2	2	3	0	7	27
SIS 320x240 (125MHz)	9	9	20	0	38	58
SIS 640x480 (20MHz)	4	3	7	0	14	34
SIS 640x480 (125MHz)	12	12	17	0	41	61
FDI 320x240 (20MHz)	2	2	3	12	19	39
FDI 320x240 (104MHz)	12	14	17	34	77	97

Figure 5.10 shows a visual comparison of the intermediate stages of the algorithm on the software and the analog section of the SIS. Figure 5.10(a) shows the image input, taken from IR security footage in the OSU database, which shows two pedestrians crossing the street. Figures 5.10(b) and (c) show the absolute frame-difference and thresholding computed by the software, and Figs. 5.10(d) and (e) show the same stages of the algorithm output by the smart pixel array and A-THR module in the SIS. The figure shows that both implementations produces visually similar results, although the SIS output loses resolution, mainly due to the reduction in integration time.

Figure 5.11 shows a visual comparison of the intermediate stages of the algorithm on the software baseline implementation and the digital coprocessor. Because the two implementations receive a single-bit pixel image as input and the algorithm uses integer arithmetic only, they can produce identical results from the same images. However, the software and hardware implementations receive different inputs as shown in Figs. 5.10(c) and (e). As a result, there are

Table 5.10: Databases used to test the performance of the proposed algorithm.

Database	Spectrum	Image size	Number of sequences	Total number of images
OSU Thermal pedestrian database [136]	Thermal IR	360×240	10	284
Terravic Motion IR database [137]	Thermal IR	320×240	18	23355

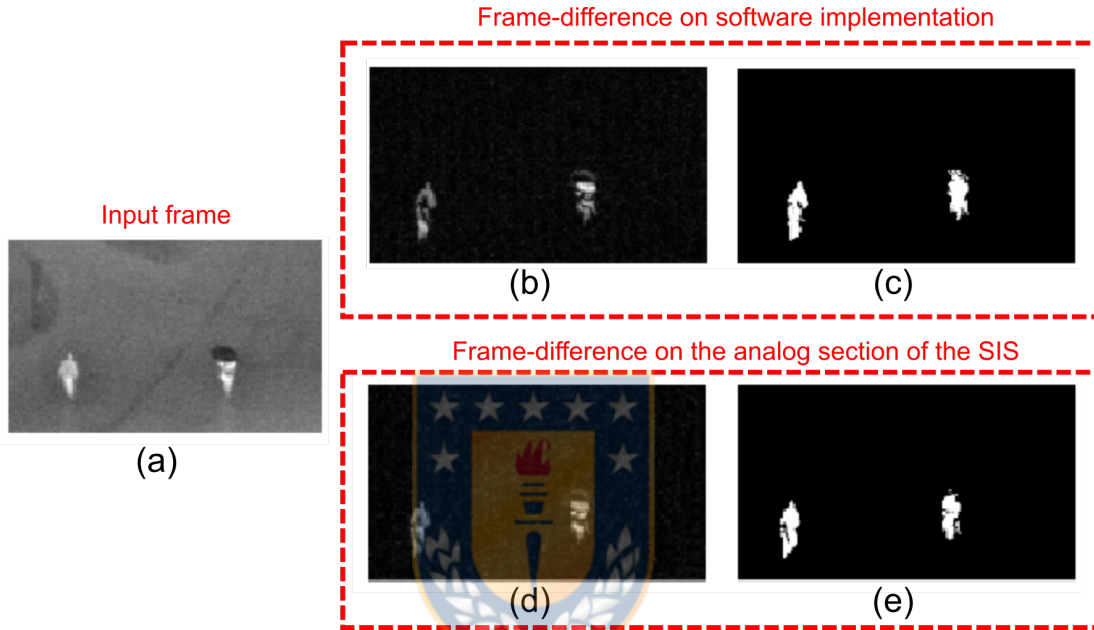


Fig. 5.10: Visual comparison of the intermediate stages of the algorithm on the software and analog section of the SIS: (a) input frame, (b) frame-difference computed by the software, (c) software output after thresholding, (d) smart-pixel array output in frame-difference mode, and (e) A-THR output in the SIS.

small differences in the image opening output (Figs. 5.11(a) and (c)), which leads to differences in the bounding boxes (Figs. 5.11(b) and (d)). Figure 5.11(e) overlaps the bounding boxes produced by the two implementations on the input image of Fig. 5.10(a).

5.3.4 Comparison to Related work

We quantified the performance of the object location algorithm in the SIS implementation using the software implementation as a baseline. We used the Intersection over Union (IoU) index to estimate the accuracy of each bounding box output [138] and the average precision (AP), which measures the fraction of the objects in the image that are correctly located by the

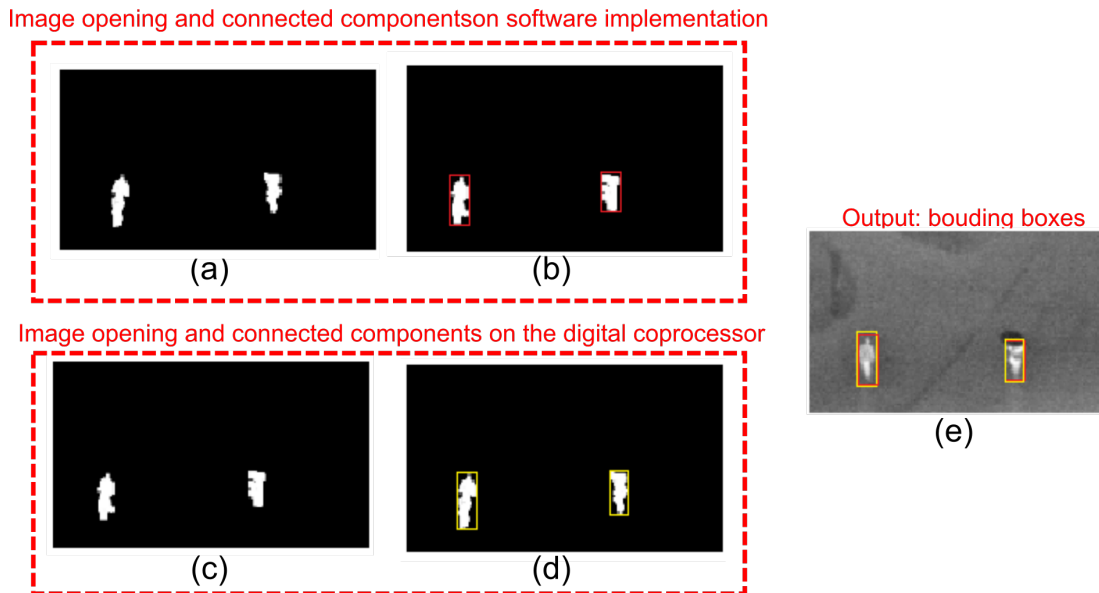


Fig. 5.11: Visual comparison of the intermediate stages of the software and digital coprocessor: (a) image opening computed by the software, (b) bounding boxes in the software, (c) image opening computed by the digital coprocessor, (d) bounding boxes output by the digital coprocessor, and (e) comparison between the outputs of the two implementations.

algorithm [138].

The IoU is defined in Equation (5.2) as:

$$\text{IoU} = \frac{\text{area}(SW \cap HW)}{\text{area}(SW \cup HW)}, \quad (5.2)$$

where SW is the ground truth given by the bounding box computed by the software implementation, and HW is the same bounding box computed by our SIS hardware implementation. The IoU equals zero when the bounding boxes computed by two implementations have no overlap, and it equals one when the bounding boxes completely match. To compute the AP, we define a set of IoU threshold values THR_{IoU} , such that the location result of the i^{th} object in the image is defined as a true positive (TP) when $IoU_i \geq THR_{IoU}$, and a false negative (FN) when $IoU_i < THR_{IoU}$. For each selected value of THR_{IoU} , the precision is computed as the ratio between the number of TP and the total number of objects ($TP + FN$) in the image. Finally, the AP of the algorithm is computed as the average between the precision values for each THR_{IoU} in the image, for all images in the dataset.

Using the OSU dataset, we computed a total of 1050 bounding boxes from the 284 input images. The average value of the IoU for all boxes is 0.94. With the Terravic dataset, we obtained a total of 65,394 bounding boxes from the 23,355 images, for an average IoU value of 0.9. To compute the AP, we used THR_{IoU} values in the range $[0.85, 0.95]$ with 0.01 increments.

Our SIS implementation of the algorithm obtained an AP of 0.92 on the OSU dataset and 0.87 on the Terravic dataset.

Table 5.11 compares the smart pixel array proposed in this work to other designs reported in the literature that implement object detection on an SIS [62, 65, 68]. We also include our own previous SIS designed for face recognition [64], which also uses an iROIC to implement pixel-level operations.

The SIS presented in [65] detects objects using pixel-level processing to compute HOG features in an 8×8 -pixel window. The processing circuits reduce the fill factor to 19%. The rest of the object detection algorithm is performed in a digital coprocessor and achieves an AP of 0.84. To improve the fill factor, the SISs in [62, 68] move most or all the computation to the column level or to a coprocessor external to the imager. The SIS presented in [62] implements motion detection only to activate the digital coprocessor that performs object detection. The SIS combines pixel- and column-level processing to implement motion detection, and achieves a fill factor of 30% despite sharing capacitors between horizontally adjacent pixels. The coprocessor achieves an AP of 0.94. The SIS presented in [68] uses a digital coprocessor that operates at the column level, using an ADC for each column. Although it adds no additional circuitry at the pixel level, the die area used by the ADCs and coprocessors limits the fill factor to 60%. The digital coprocessor achieves an AP between 0.7 and 0.87, depending on the type of object detected.

Compared to works discussed above, our SIS achieves a frame rate that is significantly higher than those reported in the literature. This is mainly due to the parallelism exploited by our design at the pixel level and the fact that our column-level circuits have a single-bit digital output, which improves the readout time. Table 5.11 also shows that our fill factor is higher than those reported in the related work when using comparable CMOS processes. The main reason for this is that our SIS uses iROICs at the pixel level to compute the frame differences, which only add a capacitor and six extra switches to the conventional integration circuit. Finally, it is important to note that our design uses a CTIA to perform integration, which allow us to operate in the IR spectrum and low-light environments. The works reported in [62, 65, 68] only operate in the visible spectrum, but this allows them to use simpler pixel architectures with smaller die area.

The final column of Table 5.11 reports our own previous SIS [64] designed for face recognition, which uses an iROIC approach similar to this work. In consequence, the design achieves a similar fill factor, with slightly less area overhead because it uses only four switches and one capacitor per pixel. However, its maximum frame rate is significantly lower because it requires multiple

Table 5.11: Comparison of our smart-pixel design for object detection to other circuits in the literature.

	This Work		[65]	[62]	[68]	[64]	[64]
Technology (μm)	0.35	0.18	0.18	0.18	0.13	0.35	0.18
Array size (pixels)	320×240	320×240	256×256	256×256	320×240	150×80	150×80
Pixel pitch (μm)	32×32	32×32	31×31	5.9×5.9	5×5	32×32	32×32
Fill Factor (%)	28	74	19	30	60	34	76
Power (μw)	8.25 (pixel)	-	2.18 (array)	51.1 (array)	229 (array)	-	-
Type of integrator	CTIA	CTIA	OTA + 2 CAP	5T + 1 CAP	4T	CTIA	CTIA
Tested spectrum	IR	IR	Visible	Visible	Visible	Visible IR/NIR	Visible IR/NIR
AP	0.87 - 0.92	-	0.84	0.94	0.7 - 0.87	-	-
SIS fps	3846	-	30	30	15 (207 max)	556	-

reads per pixel to compute the features of the image at the column level.



6. Conclusions

This report has presented two SIS architectures, one for face recognition and other for motion-based object detection. The SIS architecture for face recognition uses local gradients to extract image features based on a lightweight version of LBP. The analog smart pixel sensor computes spatial gradients in the image in parallel during photocurrent integration, and can be configured to output the regular pixel value or the local gradients. A digital coprocessor computes a modified version of Ahonen's algorithm, where LDA is used to reduce the feature space dimensions and improve class separability. The SIS for object detection uses a smart-pixel array with local memory to compute frame-differences in the analog domain during pixel-current integration with high parallelism. It also uses an analog comparator and a digital coprocessor to compute image opening and connected components to detect objects from the frame-difference output of the smart-pixel array.

The results of this work show that using a heterogeneous smart camera architecture can distribute the computation of the algorithm between a smart-pixel array and a digital coprocessor. Using analog circuitry to process on the smart-pixel array simultaneously can affect the results of the algorithms but increases the processing framerate and reduces the overall power. As a drawback, a penalty on the fill factor can decrease the effective area for light capturing, but depending on the application, this penalty is acceptable considering the benefits. Compared to traditional software implementation and programmable hardware (such as FPGAs), using smart pixels that operate simultaneously greatly increases the algorithm's parallelism being exploited. The evaluated metrics show that the processing results are still competitive with those obtained using traditional software and hardware implementations. The iROIC presented also contributes to increasing the performance of the SISs, thanks to its computation process during the capture time. Moreover, column-level circuits, such as analog thresholding or the RLBP generator, further reduce computing time.

To design a heterogeneous intelligent camera, it is important to simultaneously consider the architecture design and the algorithm's stages. This way, the resulting design can naturally implement the desired computation simultaneously along all the smart-pixel array. The simultaneous consideration of the architecture and the algorithm could include some mathematical simplifications and some algorithm's adaptation. Generalizing the pixel architecture to interface the smart-pixel array with a digital coprocessor is an essential step in the design process because

it could lead to further parallelism at the column level.

On the face detection SIS it is possible to summarize the following. Post-layout simulations of an array of 150×80 pixels of $32 \mu\text{m} \times 32 \mu\text{m}$ show that the array can deliver up to 556 frames per second. Modifying the integration readout circuit to compute the local gradients allows to extract local features with a small impact on fill factor. The digital coprocessor, implemented on a Xilinx XC7Z020 FPGA, can classify a face image in $94 \mu\text{s}$, or 10,638 images per second, while consuming 71 mW of power. Several techniques were used to reduce on-chip resource utilization, such as storing the LDA coefficients on external memory, and simultaneously building the RLBP histograms and mapping them to the LDA subspace to avoid computing matrix-vector multiplications. As a result, the coprocessor uses less than 10% of the slice LUTs of the FPGA, less than 2% of the on-chip block memory, and less than 3% of the multipliers.

When classifying faces using different databases, it is possible to observe that the proposed algorithm outperforms other methods in the literature, except when there are large variations in illumination between the training and test datasets. These variations are significantly smaller in IR images, for which the face recognition smart pixel has been designed. The results also show that replacing conventional LBP with the proposed RLBP still captures sufficient texture information to perform face classification with a small degradation in accuracy.

On the object detection SIS it is possible to summarize the following. Computing the frame-differences on the smart-pixel array eliminates the need for a frame buffer in the digital coprocessor. Indeed, the results show that the coprocessor in the proposed SIS does not use on-chip memory blocks on the FPGA, while a fully-digital implementation of the algorithm requires 19 memory blocks for 320×240 -pixel images, and 75 blocks for a 640×480 -pixel input. The latter can not be implemented on the entry-level XC7A35T FPGA, which features only 50 memory blocks. As a result, the digital coprocessor attached to the SIS also achieves a higher maximum clock frequency, and therefore a higher frame rate, than the fully digital implementation of the algorithm.

The results show that, using a $32 \mu\text{m} \times 32 \mu\text{m}$ pixel, the proposed design reduces the fill factor on the $0.18 \mu\text{m}$ process from 86.3% to 74% compared to a traditional CTIA-based imager. This small impact on the fill factor is mainly due to the addition of switches that control the current flow during integration. Because the integration time is reduced by 50% in frame-difference mode, the resolution of the output images reduced. However, the circuit can still detect objects with a mean accuracy of 0.92 over 28 video sequences from two thermal IR databases, compared to a reference software implementation. When the presented SIS is compared with a fully digital implementation, it is possible to note that the proposed circuits

has lower area utilization and power consumption.

When we use integration capacitors as double-buffer memory to compute frame differences, we reduce the penalty on the fill factor compared to circuits that operate with readout-circuit output. Furthermore, although our smart pixel effectively uses half of the integration time, which could reduce the signal-to-noise ratio, our results are comparable to a software implementation of the motion-based object location algorithm. The on-imager computation of the proposed object detection SISs is convenient in contexts where privacy is required, where it eliminates the need to continuously transmit video information over a communication channel. As an example, the object detection SIS can deliver an alarm only when objects in motion are detected. Another example is the use of the proposed object detection SIS paired with a high-resolution camera where the SIS could detect objects based on motion and send the bounding boxes to an external controller, which could use them to activate the capture of that portion of the high resolution image. The proposed SISs have low power consumption and low area utilization, making them suitable for mobile devices and portable systems. Although the CTIA integrator used in the smart pixel is larger than alternative readout circuits, it is suitable for IR and low-light imagers. Computing local differences during photocurrent integration minimizes the impact on circuit area and fill factor, even though by cutting the integration time in half, it may reduce the signal-to-noise ratio of the image sensor in face-recognition mode.

6.1 Future work

Thanks to the work presented on this report, it is possible to continue research in the field of SIS as it was explored in this thesis. First, thanks to the area availability on each design it is possible to join their architecture into a single readout circuit to perform local gradients or temporal differences on-pixel. Then, it is possible to unify the digital coprocessors into a single SIS to work as a conventional imager, a face-recognition or an object detection system. From that point, it is possible to further expand the architecture and generate a programming model to easily program the multi-purpose smart pixel. This programming model could include the selection of the integration capacitor, the polarity of the integration on each, the photodetector, and their respective integration time ratio.

Other related research lines could include the use of the proposed smart pixels as a base for other computer vision algorithms that need to compute local gradients or temporal changes on-pixel. Such applications includes local kernels for edge detection, local filters for noise reduction, and temporal statistics such as the mobile mean estimation. Further, it is possible to think of

other smart pixel architectures that use the same technique of adding or subtracting analog values on-pixel. These could include smart pixels for non-uniformity correction on infrared sensors, such as neural network based algorithms or two point calibration.



Bibliography

- [1] N. Sanil, P. A. N. venkat, V. Rakesh, R. Mallapur, and M. R. Ahmed, “Deep Learning Techniques for Obstacle Detection and Avoidance in Driverless Cars,” in *2020 International Conference on Artificial Intelligence and Signal Processing (AISP)*. Amaravati, India: IEEE, Jan. 2020, pp. 1–4. [Online]. Available: <https://ieeexplore.ieee.org/document/9073155/>
- [2] H. M. Thakurdesai and J. V. Aghav, “Computer Vision Based Position and Speed Estimation for Accident Avoidance in Driverless Cars,” in *ICT Systems and Sustainability*, M. Tuba, S. Akashe, and A. Joshi, Eds. Singapore: Springer Singapore, 2020, vol. 1077, pp. 435–443. [Online]. Available: http://link.springer.com/10.1007/978-981-15-0936-0_47
- [3] Y. Zhu, J. Yang, X. Deng, C. Xiao, and W. An, “Infrared Pedestrian Detection Based on Attention Mechanism,” *Journal of Physics: Conference Series*, vol. 1634, no. 1, p. 012032, Sep. 2020. [Online]. Available: <https://iopscience.iop.org/article/10.1088/1742-6596/1634/1/012032>
- [4] H.-J. Kwon and S.-H. Lee, “Visible and Near-Infrared Image Acquisition and Fusion for Night Surveillance,” *Chemosensors*, vol. 9, no. 4, p. 75, Apr. 2021. [Online]. Available: <https://www.mdpi.com/2227-9040/9/4/75>
- [5] M. Salhaoui, J. C. Molina-Molina, A. Guerrero-González, M. Arioua, and F. J. Ortiz, “Autonomous Underwater Monitoring System for Detecting Life on the Seabed by Means of Computer Vision Cloud Services,” *Remote Sensing*, vol. 12, no. 12, p. 1981, Jun. 2020. [Online]. Available: <https://www.mdpi.com/2072-4292/12/12/1981>
- [6] V. Kakani, V. H. Nguyen, B. P. Kumar, H. Kim, and V. R. Pasupuleti, “A critical review on computer vision and artificial intelligence in food industry,” *Journal of Agriculture and Food Research*, vol. 2, p. 100033, Dec. 2020. [Online]. Available: <https://linkinghub.elsevier.com/retrieve/pii/S2666154320300144>
- [7] W. Khan, A. Hussain, K. Kuru, and H. Al-askar, “Pupil Localisation and Eye Centre Estimation Using Machine Learning and Computer Vision,” *Sensors*, vol. 20, no. 13, p. 3785, Jul. 2020. [Online]. Available: <https://www.mdpi.com/1424-8220/20/13/3785>

- [8] G. Sikander and S. Anwar, "Driver Fatigue Detection Systems: A Review," *IEEE Transactions on Intelligent Transportation Systems*, vol. 20, no. 6, pp. 2339–2352, Jun. 2019. [Online]. Available: <https://ieeexplore.ieee.org/document/8482470/>
- [9] Z. Rui and Z. Yan, "A Survey on Biometric Authentication: Toward Secure and Privacy-Preserving Identification," *IEEE Access*, vol. 7, pp. 5994–6009, 2019. [Online]. Available: <https://ieeexplore.ieee.org/document/8590812/>
- [10] M. Sharif, M. Raza, J. H. Shah, M. Yasmin, and S. L. Fernandes, "An Overview of Biometrics Methods," in *Handbook of Multimedia Information Security: Techniques and Applications*, A. K. Singh and A. Mohan, Eds. Cham: Springer International Publishing, 2019, pp. 15–35. [Online]. Available: https://doi.org/10.1007/978-3-030-15887-3_2
- [11] I. Adjabi, A. Ouahabi, A. Benzaoui, and A. Taleb-Ahmed, "Past, Present, and Future of Face Recognition: A Review," *Electronics*, vol. 9, no. 8, p. 1188, jul 2020. [Online]. Available: <https://www.mdpi.com/2079-9292/9/8/1188>
- [12] M. O. Oloyede, G. P. Hancke, and H. C. Myburgh, "A review on face recognition systems: recent approaches and challenges," *Multimedia Tools and Applications*, vol. 79, no. 37–38, pp. 27 891–27 922, oct 2020. [Online]. Available: <http://link.springer.com/10.1007/s11042-020-09261-2>
- [13] V. D. Ambeth Kumar, S. Ramya, H. Divakar, and G. Kumutha Rajeswari, "A Survey on Face Recognition in Video Surveillance," in *Proceedings of the International Conference on ISMAC in Computational Vision and Bio-Engineering 2018 (ISMAC-CVB)*, D. Pandian, X. Fernando, Z. Baig, and F. Shi, Eds. Cham: Springer International Publishing, 2019, vol. 30, pp. 699–708, series Title: Lecture Notes in Computational Vision and Biomechanics. [Online]. Available: http://link.springer.com/10.1007/978-3-030-00665-5_69
- [14] F. A. Pujol, M. J. Pujol, C. Rizo-Maestre, and M. Pujol, "Entropy-based face recognition and spoof detection for security applications," *Sustainability*, vol. 12, no. 1, 2020. [Online]. Available: <https://www.mdpi.com/2071-1050/12/1/85>
- [15] M. G. Galterio, S. A. Shavit, and T. Hayajneh, "A review of facial biometrics security for smart devices," *Computers*, vol. 7, no. 3, 2018. [Online]. Available: <https://www.mdpi.com/2073-431X/7/3/37>
- [16] S. Pandey, V. Chouhan, R. P. Mahapatra, D. Chhettri, and H. Sharma, "Real-time safety and surveillance system using facial recognition mechanism," in *Intelligent Computing*

- and Applications*, S. S. Dash, S. Das, and B. K. Panigrahi, Eds. Singapore: Springer Singapore, 2021, pp. 497–506.
- [17] N. T. Son, B. N. Anh, T. Q. Ban, L. P. Chi, B. D. Chien, D. X. Hoa, L. V. Thanh, T. Q. Huy, L. D. Duy, M. Hassan Raza Khan *et al.*, “Implementing cctv-based attendance taking support system using deep face recognition: A case study at fpt polytechnic college,” *Symmetry*, vol. 12, no. 2, p. 307, 2020.
- [18] M. Muthumari, N. K. Sah, R. Raj, and J. Saharia, “Arduino based Auto Door unlock control system by Android mobile through Bluetooth and Wi-Fi,” in *2018 IEEE International Conference on Computational Intelligence and Computing Research (ICICR)*. Madurai, India: IEEE, dec 2018, pp. 1–4. [Online]. Available: <https://ieeexplore.ieee.org/document/8782297/>
- [19] C. Morikawa, M. Kobayashi, M. Satoh, Y. Kuroda, T. Inomata, H. Matsuo, T. Miura, and M. Hilaga, “Image and video processing on mobile devices: a survey,” *The Visual Computer*, vol. 37, no. 12, pp. 2931–2949, Dec. 2021. [Online]. Available: <https://link.springer.com/10.1007/s00371-021-02200-8>
- [20] A. Das, C. Galdi, H. Han, R. Ramachandra, J.-L. Dugelay, and A. Dantcheva, “Recent Advances in Biometric Technology for Mobile Devices,” in *2018 IEEE 9th International Conference on Biometrics Theory, Applications and Systems (BTAS)*. Redondo Beach, CA, USA: IEEE, oct 2018, pp. 1–11. [Online]. Available: <https://ieeexplore.ieee.org/document/8698587/>
- [21] L. Liu, H. Li, and M. Gruteser, “Edge Assisted Real-time Object Detection for Mobile Augmented Reality,” in *The 25th Annual International Conference on Mobile Computing and Networking*. Los Cabos Mexico: ACM, Aug. 2019, pp. 1–16. [Online]. Available: <https://dl.acm.org/doi/10.1145/3300061.3300116>
- [22] R. J. Wang, X. Li, and C. X. Ling, “Pelee: A Real-Time Object Detection System on Mobile Devices,” 2018, publisher: arXiv Version Number: 3. [Online]. Available: <https://arxiv.org/abs/1804.06882>
- [23] B. Chen, G. Ghiasi, H. Liu, T.-Y. Lin, D. Kalenichenko, H. Adam, and Q. V. Le, “MnasFPN: Learning Latency-Aware Pyramid Architecture for Object Detection on Mobile Devices,” in *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. Seattle, WA, USA: IEEE, Jun. 2020, pp. 13 604–13 613. [Online]. Available: <https://ieeexplore.ieee.org/document/9156863/>

- [24] R. N. Mayo and P. Ranganathan, “Energy Consumption in Mobile Devices: Why Future Systems Need Requirements-Aware Energy Scale-Down,” in *Power-Aware Computer Systems*, B. Falsafi and T. N. VijayKumar, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2005, vol. 3164, pp. 26–40, series Title: Lecture Notes in Computer Science. [Online]. Available: http://link.springer.com/10.1007/978-3-540-28641-7_3
- [25] A. HajiRassouliha, A. J. Taberner, M. P. Nash, and P. M. Nielsen, “Suitability of recent hardware accelerators (DSPs, FPGAs, and GPUs) for computer vision and image processing algorithms,” *Signal Processing: Image Communication*, vol. 68, pp. 101–119, Oct. 2018. [Online]. Available: <https://linkinghub.elsevier.com/retrieve/pii/S0923596518303606>
- [26] M. Khairy, A. G. Wassal, and M. Zahran, “A survey of architectural approaches for improving GPGPU performance, programmability and heterogeneity,” *Journal of Parallel and Distributed Computing*, vol. 127, pp. 65–88, May 2019. [Online]. Available: <https://linkinghub.elsevier.com/retrieve/pii/S0743731518308669>
- [27] X. Yin, L. Chen, X. Zhang, and Z. Gao, “Object Detection Implementation and Optimization on Embedded GPU System,” in *2018 IEEE International Symposium on Broadband Multimedia Systems and Broadcasting (BMSB)*. Valencia: IEEE, Jun. 2018, pp. 1–5. [Online]. Available: <https://ieeexplore.ieee.org/document/8436848/>
- [28] S. Pirbhulal, H. Zhang, W. Wu, S. C. Mukhopadhyay, and Y. T. Zhang, “Heartbeats based biometric random binary sequences generation to secure wireless body sensor networks,” *IEEE Transactions on Biomedical Engineering*, vol. 65, no. 12, pp. 2751–2759, 2018.
- [29] B. Kaur, D. Singh, and P. P. Roy, “Age and gender classification using brain-computer interface,” *Neural Computing and Applications*, vol. 31, no. 10, pp. 5887–5900, oct 2019. [Online]. Available: <http://link.springer.com/10.1007/s00521-018-3397-1>
- [30] Shayan Taheri and Jiann-Shiun Yuan, “A Cross-Layer Biometric Recognition System for Mobile IoT Devices,” *Electronics*, vol. 7, no. 2, p. 26, feb 2018. [Online]. Available: <http://www.mdpi.com/2079-9292/7/2/26>
- [31] K. Bong, S. Choi, C. Kim, D. Han, and H.-J. Yoo, “A Low-Power Convolutional Neural Network Face Recognition Processor and a CIS Integrated With Always-on Face Detector,” *IEEE Journal of Solid-State Circuits*, vol. 53, no. 1, pp. 115–123, jan 2018. [Online]. Available: <http://ieeexplore.ieee.org/document/8197364/>

- [32] C. Kim, K. Bong, I. Hong, K. Lee, S. Choi, and H.-J. Yoo, "An ultra-low-power and mixed-mode event-driven face detection SoC for always-on mobile applications," in *ESSCIRC 2017 - 43rd IEEE European Solid State Circuits Conference*. Leuven: IEEE, sep 2017, pp. 255–258. [Online]. Available: <http://ieeexplore.ieee.org/document/8094574/>
- [33] J.-H. Kim, C. Kim, K. Kim, and H.-J. Yoo, "An Ultra-Low-Power Analog-Digital Hybrid CNN Face Recognition Processor Integrated with a CIS for Always-on Mobile Devices," in *2019 IEEE International Symposium on Circuits and Systems (ISCAS)*. Sapporo, Japan: IEEE, may 2019, pp. 1–5. [Online]. Available: <https://ieeexplore.ieee.org/document/8702698/>
- [34] J. Ohta, *Smart CMOS image sensors and applications*, 2nd ed., ser. Optical science and engineering. Boca Raton: CRC Press, 2020.
- [35] J. Yang, C. Shi, Z. Cao, Y. Han, L. Liu, and N. Wu, "Smart image sensing system," in *2013 IEEE SENSORS*. Baltimore, MD, USA: IEEE, nov 2013, pp. 1–4. [Online]. Available: <http://ieeexplore.ieee.org/document/6688261/>
- [36] M. Jin, H. Noh, M. Song, and S. Y. Kim, "Design of an Edge-Detection CMOS Image Sensor with Built-in Mask Circuits," *Sensors*, vol. 20, no. 13, p. 3649, jun 2020. [Online]. Available: <https://www.mdpi.com/1424-8220/20/13/3649>
- [37] C. Yin and C.-C. Hsieh, "A 0.5V 34.4uW 14.28kfps 105dB smart image sensor with array-level analog signal processing," in *2013 IEEE Asian Solid-State Circuits Conference (A-SSCC)*. Singapore, Singapore: IEEE, nov 2013, pp. 97–100. [Online]. Available: <http://ieeexplore.ieee.org/document/6690991/>
- [38] J. Choi, S. Lee, Y. Son, and S. Y. Kim, "Design of an Always-On Image Sensor Using an Analog Lightweight Convolutional Neural Network," *Sensors*, vol. 20, no. 11, p. 3101, may 2020. [Online]. Available: <https://www.mdpi.com/1424-8220/20/11/3101>
- [39] S. Xie, A. A. Prouza, and A. Theuwissen, "A cmos-imager-pixel-based temperature sensor for dark current compensation," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 67, no. 2, pp. 255–259, 2020.
- [40] T. Zhou, J. Zhao, Y. He, B. Jiang, and Y. Su, "A readout integrated circuit (roic) employing self-adaptive background current compensation technique for infrared focal plane array (irfpa)," *Infrared Physics & Technology*, vol. 90, pp. 122–132, 2018. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S1350449517307879>

- [41] K. Lee, S. Park, S.-Y. Park, J. Cho, and E. Yoon, "A 272.49 pj/pixel cmos image sensor with embedded object detection and bio-inspired 2d optic flow generation for nano-air-vehicle navigation," in *2017 Symposium on VLSI Circuits*. IEEE, 2017, pp. C294–C295.
- [42] X. Zhong, Q. Yu, A. Bermak, C.-Y. Tsui, and M.-K. Law, "A 2pj/pixel/direction mimo processing based cmos image sensor for omnidirectional local binary pattern extraction and edge detection," in *2018 IEEE Symposium on VLSI Circuits*. IEEE, 2018, pp. 247–248.
- [43] J. Choi, S. Park, J. Cho, and E. Yoon, "A 3.4- μ W Object-Adaptive CMOS Image Sensor With Embedded Feature Extraction Algorithm for Motion-Triggered Object-of-Interest Imaging," *IEEE Journal of Solid-State Circuits*, vol. 49, no. 1, pp. 289–300, jan 2014. [Online]. Available: <http://ieeexplore.ieee.org/document/6642143/>
- [44] T.-H. Hsu, Y.-R. Chen, R.-S. Liu, C.-C. Lo, K.-T. Tang, M.-F. Chang, and C.-C. Hsieh, "A 0.5-V Real-Time Computational CMOS Image Sensor With Programmable Kernel for Feature Extraction," *IEEE Journal of Solid-State Circuits*, pp. 1–1, 2020. [Online]. Available: <https://ieeexplore.ieee.org/document/9250500/>
- [45] N. Massari and M. Gottardi, "A 100 db dynamic-range cmos vision sensor with programmable image processing and global feature extraction," *IEEE Journal of Solid-State Circuits*, vol. 42, no. 3, pp. 647–657, 2007.
- [46] W. Zhang, Q. Fu, and N.-J. Wu, "A Programmable Vision Chip Based on Multiple Levels of Parallel Processors," *IEEE Journal of Solid-State Circuits*, vol. 46, no. 9, pp. 2132–2147, sep 2011. [Online]. Available: <http://ieeexplore.ieee.org/document/5936648/>
- [47] J. Hasler, "Analog architecture complexity theory empowering ultra-low power configurable analog and mixed mode soc systems," *Journal of Low Power Electronics and Applications*, vol. 9, no. 1, 2019. [Online]. Available: <https://www.mdpi.com/2079-9268/9/1/4>
- [48] A. J. Sanchez-Fernandez, L. F. Romero, D. Peralta, M. A. Medina-Pérez, Y. Saeys, F. Herrera, and S. Tabik, "Asynchronous processing for latent fingerprint identification on heterogeneous cpu-gpu systems," *IEEE Access*, vol. 8, pp. 124 236–124 253, 2020.
- [49] S. Zhang, X. Wang, Z. Lei, and S. Z. Li, "Faceboxes: A cpu real-time and accurate unconstrained face detector," *Neurocomputing*, vol. 364, pp. 297–309, 2019. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S0925231219310719>

- [50] M. T. Ahmed and S. Sinha, "Design and development of efficient face recognition architecture using neural network on fpga," in *2018 Second International Conference on Intelligent Computing and Control Systems (ICICCS)*, 2018, pp. 905–909.
- [51] G. Mahale, H. Mahale, A. Goel, S. K. Nandy, S. Bhattacharya, and R. Narayan, "Hardware solution for real-time face recognition," in *2015 28th International Conference on VLSI Design*, 2015, pp. 81–86.
- [52] X. Qu, T. Wei, C. Peng, and P. Du, "A fast face recognition system based on deep learning," in *2018 11th International Symposium on Computational Intelligence and Design (ISCID)*, vol. 01, 2018, pp. 289–292.
- [53] T. Šušteršič and A. Peulić, "Implementation of Face Recognition Algorithm on Field Programmable Gate Array (FPGA)," *Journal of Circuits, Systems and Computers*, vol. 28, no. 08, p. 1950129, Jul. 2019. [Online]. Available: <https://www.worldscientific.com/doi/abs/10.1142/S0218126619501299>
- [54] J. E. Soto and M. Figueroa, "An embedded face-classification system for infrared images on an FPGA," in *Electro-Optical and Infrared Systems: Technology and Applications XI*, D. A. Huckridge and R. Ebert, Eds., vol. 9249, International Society for Optics and Photonics. SPIE, 2014, pp. 159 – 170. [Online]. Available: <https://doi.org/10.1117/12.2067488>
- [55] R. Zhao, X. Niu, Y. Wu, W. Luk, and Q. Liu, "Optimizing CNN-Based Object Detection Algorithms on Embedded FPGA Platforms," in *Applied Reconfigurable Computing*, S. Wong, A. C. Beck, K. Bertels, and L. Carro, Eds. Cham: Springer International Publishing, 2017, vol. 10216, pp. 255–267, series Title: Lecture Notes in Computer Science. [Online]. Available: http://link.springer.com/10.1007/978-3-319-56258-2_22
- [56] H. Fan, S. Liu, M. Ferianc, H.-C. Ng, Z. Que, S. Liu, X. Niu, and W. Luk, "A Real-Time Object Detection Accelerator with Compressed SSDLite on FPGA," in *2018 International Conference on Field-Programmable Technology (FPT)*. Naha, Okinawa, Japan: IEEE, Dec. 2018, pp. 14–21. [Online]. Available: <https://ieeexplore.ieee.org/document/8742299/>
- [57] D. T. Nguyen, T. N. Nguyen, H. Kim, and H.-J. Lee, "A High-Throughput and Power-Efficient FPGA Implementation of YOLO CNN for Object Detection," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 27, no. 8, pp. 1861–1873, Aug. 2019. [Online]. Available: <https://ieeexplore.ieee.org/document/8678682/>

- [58] A. Sharma, V. Singh, and A. Rani, "Implementation of CNN on Zynq based FPGA for Real-time Object Detection," in *2019 10th International Conference on Computing, Communication and Networking Technologies (ICCCNT)*. Kanpur, India: IEEE, Jul. 2019, pp. 1–7. [Online]. Available: <https://ieeexplore.ieee.org/document/8944792/>
- [59] N. Zhang, X. Wei, H. Chen, and W. Liu, "FPGA Implementation for CNN-Based Optical Remote Sensing Object Detection," *Electronics*, vol. 10, no. 3, p. 282, Jan. 2021. [Online]. Available: <https://www.mdpi.com/2079-9292/10/3/282>
- [60] X. Long, S. Hu, Y. Hu, Q. Gu, and I. Ishii, "An FPGA-Based Ultra-High-Speed Object Detection Algorithm with Multi-Frame Information Fusion," *Sensors*, vol. 19, no. 17, p. 3707, Aug. 2019. [Online]. Available: <https://www.mdpi.com/1424-8220/19/17/3707>
- [61] H. Nakahara, H. Yonekawa, and S. Sato, "An object detector based on multiscale sliding window search using a fully pipelined binarized CNN on an FPGA," in *2017 International Conference on Field Programmable Technology (ICFPT)*. Melbourne, VIC: IEEE, Dec. 2017, pp. 168–175. [Online]. Available: <http://ieeexplore.ieee.org/document/8280135/>
- [62] J. Choi, S. Park, J. Cho, and E. Yoon, "A 3.4- μW Object-Adaptive CMOS Image Sensor With Embedded Feature Extraction Algorithm for Motion-Triggered Object-of-Interest Imaging," *IEEE Journal of Solid-State Circuits*, vol. 49, no. 1, pp. 289–300, Jan. 2014. [Online]. Available: <http://ieeexplore.ieee.org/document/6642143/>
- [63] R. Hameed, W. Qadeer, M. Wachs, O. Azizi, A. Solomatnikov, B. C. Lee, S. Richardson, C. Kozyrakis, and M. Horowitz, "Understanding sources of inefficiency in general-purpose chips," in *Proceedings of the 37th annual international symposium on Computer architecture*, 2010, pp. 37–47.
- [64] W. Valenzuela, J. E. Soto, P. Zarkesh-Ha, and M. Figueroa, "Face Recognition on a Smart Image Sensor Using Local Gradients," *Sensors*, vol. 21, no. 9, p. 2901, Apr. 2021. [Online]. Available: <https://www.mdpi.com/1424-8220/21/9/2901>
- [65] K. Lee, S. Park, S.-Y. Park, J. Cho, and E. Yoon, "A 272.49 pJ/pixel CMOS image sensor with embedded object detection and bio-inspired 2D optic flow generation for nano-air-vehicle navigation," in *2017 Symposium on VLSI Circuits*. IEEE, 2017, pp. C294–C295.
- [66] P. Zarkesh-Ha, "An intelligent readout circuit for infrared multispectral remote sensing," in *2014 IEEE 57th International Midwest Symposium on Circuits and Systems (MWS-CAS)*. IEEE, 2014, pp. 153–156.

- [67] M. Gottardi and M. Lecca, “A 64×64 Pixel Vision Sensor for Local Binary Pattern Computation,” *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 66, no. 5, pp. 1831–1839, 2019.
- [68] C. Young, A. Omid-Zohoor, P. Lajevardi, and B. Murmann, “A Data-Compressive 1.5/2.75-bit Log-Gradient QVGA Image Sensor With Multi-Scale Readout for Always-On Object Detection,” *IEEE Journal of Solid-State Circuits*, vol. 54, no. 11, pp. 2932–2946, Nov. 2019. [Online]. Available: <https://ieeexplore.ieee.org/document/8844721/>
- [69] M.-S. Shin, J.-B. Kim, M.-K. Kim, Y.-R. Jo, and O.-K. Kwon, “A 1.92-megapixel CMOS image sensor with column-parallel low-power and area-efficient SA-ADCs,” *IEEE Transactions on Electron Devices*, vol. 59, no. 6, pp. 1693–1700, 2012, publisher: IEEE.
- [70] X. Zhong, Q. Yu, A. Bermak, C.-Y. Tsui, and M.-K. Law, “A 2pJ/pixel/direction MIMO processing based CMOS image sensor for omnidirectional local binary pattern extraction and edge detection,” in *2018 IEEE Symposium on VLSI Circuits*. IEEE, 2018, pp. 247–248.
- [71] M. Gottardi and M. Lecca, “A 64×64 pixel vision sensor for local binary pattern computation,” *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 66, no. 5, pp. 1831–1839, 2019.
- [72] S. Jia, G. Guo, Z. Xu, and Q. Wang, “Face presentation attack detection in mobile scenarios: A comprehensive evaluation,” *Image and Vision Computing*, vol. 93, p. 103826, 2020. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S0262885619304196>
- [73] D. Popa, S. Z. Ali, R. Hopper, Y. Dai, and F. Udrea, “Smart cmos mid-infrared sensor array,” *Opt. Lett.*, vol. 44, no. 17, pp. 4111–4114, Sep 2019. [Online]. Available: <http://ol.osa.org/abstract.cfm?URI=ol-44-17-4111>
- [74] H. Jo and W.-Y. Kim, “Nir reflection augmentation for deeplearning-based nir face recognition,” *Symmetry*, vol. 11, no. 10, 2019. [Online]. Available: <https://www.mdpi.com/2073-8994/11/10/1234>
- [75] G. Hermosilla Vigneau, J. L. Verdugo, G. Farias Castro, F. Pizarro, and E. Vera, “Thermal face recognition under temporal variation conditions,” *IEEE Access*, vol. 5, pp. 9663–9672, 2017.

- [76] Y. Kortli, M. Jridi, A. Al Falou, and M. Atri, "Face Recognition Systems: A Survey," *Sensors*, vol. 20, no. 2, p. 342, Jan. 2020. [Online]. Available: <https://www.mdpi.com/1424-8220/20/2/342>
- [77] Z. B. Lahaw, D. Essaidani, and H. Seddik, "Robust face recognition approaches using pca, ica, lda based on dwt, and svm algorithms," in *2018 41st International Conference on Telecommunications and Signal Processing (TSP)*, 2018, pp. 1–5.
- [78] L. Shi, X. Wang, and Y. Shen, "Research on 3d face recognition method based on lbp and svm," *Optik*, vol. 220, p. 165157, 2020. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S0030402620309931>
- [79] M. Ayyad and C. Khalid, "New fusion of svd and relevance weighted lda for face recognition," *Procedia Computer Science*, vol. 148, pp. 380–388, 2019, THE SECOND INTERNATIONAL CONFERENCE ON INTELLIGENT COMPUTING IN DATA SCIENCES, ICDS2018. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S1877050919300468>
- [80] K. Bonnen, B. F. Klare, and A. K. Jain, "Component-Based Representation in Automated Face Recognition," *IEEE Transactions on Information Forensics and Security*, vol. 8, no. 1, pp. 239–253, Jan. 2013. [Online]. Available: <http://ieeexplore.ieee.org/document/6341076/>
- [81] J. Ren, X. Jiang, and J. Yuan, "Relaxed local ternary pattern for face recognition," in *2013 IEEE International Conference on Image Processing*. Melbourne, Australia: IEEE, Sep. 2013, pp. 3680–3684. [Online]. Available: <http://ieeexplore.ieee.org/document/6738759/>
- [82] M. Annalakshmi, S. M. M. Roomi, and A. S. Naveedh, "A hybrid technique for gender classification with slbp and hog features," *Cluster Computing*, vol. 22, no. 1, pp. 11–20, 2019.
- [83] K. T. Islam, S. Wijewickrema, R. G. Raj, and S. O'Leary, "Street sign recognition using histogram of oriented gradients and artificial neural networks," *Journal of Imaging*, vol. 5, no. 4, p. 44, 2019.
- [84] D. Cheng, S. Tang, C. Feng *et al.*, "Extended hog-clbc for pedestrian detection [j]," *Opto-Electronic Engineering*, vol. 45, no. 8, p. 180111, 2018.
- [85] S. Gupta, K. Thakur, and M. Kumar, "2D-human face recognition using SIFT and SURF descriptors of face's feature regions," *The Visual Computer*, vol. 37, no. 3, pp. 447–456, Mar. 2021. [Online]. Available: <http://link.springer.com/10.1007/s00371-020-01814-8>

- [86] S. Setta, S. Sinha, M. Mishra, and P. Choudhury, “Real-Time Facial Recognition Using SURF-FAST,” in *Data Management, Analytics and Innovation*, N. Sharma, A. Chakrabarti, V. E. Balas, and A. M. Bruckstein, Eds. Singapore: Springer Singapore, 2022, vol. 71, pp. 505–522, series Title: Lecture Notes on Data Engineering and Communications Technologies. [Online]. Available: https://link.springer.com/10.1007/978-981-16-2937-2_32
- [87] H. Jumahong and G. Alimjan, “Face Recognition Based on Rearranged Modular Two-Dimensional Locality Preserving Projection,” *International Journal of Pattern Recognition and Artificial Intelligence*, vol. 32, no. 12, p. 1856016, Dec. 2018. [Online]. Available: <https://www.worldscientific.com/doi/abs/10.1142/S0218001418560165>
- [88] A. S. Al-Waisy, R. Qahwaji, S. Ipson, and S. Al-Fahdawi, “A multimodal deep learning framework using local feature representations for face recognition,” *Machine Vision and Applications*, vol. 29, no. 1, pp. 35–54, Jan. 2018. [Online]. Available: <http://link.springer.com/10.1007/s00138-017-0870-2>
- [89] M. Sushama and E. Rajinikanth, “Face Recognition Using DRLBP and SIFT Feature Extraction,” in *2018 International Conference on Communication and Signal Processing (ICCSP)*. Chennai: IEEE, Apr. 2018, pp. 994–999. [Online]. Available: <https://ieeexplore.ieee.org/document/8524427/>
- [90] M. Turk and A. Pentland, “Eigenfaces for Recognition,” *Journal of Cognitive Neuroscience*, vol. 3, no. 1, pp. 71–86, Jan. 1991. [Online]. Available: <https://direct.mit.edu/jocn/article/3/1/71/3025/Eigenfaces-for-Recognition>
- [91] L. Sirovich and M. Kirby, “Low-dimensional procedure for the characterization of human faces,” *Journal of the Optical Society of America A*, vol. 4, no. 3, p. 519, Mar. 1987. [Online]. Available: <https://opg.optica.org/abstract.cfm?URI=josaa-4-3-519>
- [92] M. Kirby and L. Sirovich, “Application of the Karhunen-Loeve procedure for the characterization of human faces,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 12, no. 1, pp. 103–108, Jan. 1990. [Online]. Available: <http://ieeexplore.ieee.org/document/41390/>
- [93] P. Belhumeur, J. Hespanha, and D. Kriegman, “Eigenfaces vs. Fisherfaces: recognition using class specific linear projection,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 19, no. 7, pp. 711–720, Jul. 1997. [Online]. Available: <http://ieeexplore.ieee.org/document/598228/>

- [94] J. V. Stone, "Independent component analysis: an introduction," *Trends in Cognitive Sciences*, vol. 6, no. 2, pp. 59–64, Feb. 2002. [Online]. Available: <https://linkinghub.elsevier.com/retrieve/pii/S1364661300018131>
- [95] J. Gonzalez-Lopez, S. Ventura, and A. Cano, "Distributed nearest neighbor classification for large-scale multi-label data on spark," *Future Generation Computer Systems*, vol. 87, pp. 66–82, 2018.
- [96] M. Z. N. Al-Dabagh, M. Alhabib, and F. Al-Mukhtar, "Face recognition system based on kernel discriminant analysis, k-nearest neighbor and support vector machine," *International Journal of Research and Engineering*, vol. 5, no. 3, pp. 335–338, 2018.
- [97] K. Shankar, S. Lakshmanaprabu, D. Gupta, A. Maseleno, and V. H. C. De Albuquerque, "Optimal feature-based multi-kernel svm approach for thyroid disease classification," *The journal of supercomputing*, vol. 76, no. 2, pp. 1128–1143, 2020.
- [98] S. Almabdy and L. Elrefaei, "Deep convolutional neural network-based approaches for face recognition," *Applied Sciences*, vol. 9, no. 20, p. 4397, 2019.
- [99] D. K. Jain, P. Shamsolmoali, and P. Sehdev, "Extended deep neural network for facial emotion recognition," *Pattern Recognition Letters*, vol. 120, pp. 69–74, 2019.
- [100] T. Ahonen, J. Matas, C. He, and M. Pietikäinen, "Rotation invariant image description with local binary pattern histogram fourier features," in *Scandinavian conference on image analysis*. Springer, 2009, pp. 61–70.
- [101] J. E. Soto, W. E. Valenzuela, S. Diaz, A. Saavedra, M. Figueroa, J. Ghasemi, and P. Zarkesh-Ha, "An intelligent readout integrated circuit (iROIC) with on-chip local gradient operations," in *2017 24th IEEE International Conference on Electronics, Circuits and Systems (ICECS)*. Batumi: IEEE, dec 2017, pp. 360–362. [Online]. Available: <http://ieeexplore.ieee.org/document/8292082/>
- [102] B. W. Yohanes, R. D. Airlangga, and I. Setyawan, "Real time face recognition comparison using fisherfaces and local binary pattern," in *2018 4th International Conference on Science and Technology (ICST)*. IEEE, 2018, pp. 1–5.
- [103] Z. Zou, Z. Shi, Y. Guo, and J. Ye, "Object Detection in 20 Years: A Survey," 2019, publisher: arXiv Version Number: 2. [Online]. Available: <https://arxiv.org/abs/1905.05055>

- [104] A. Toshev, B. Taskar, and K. Daniilidis, “Shape-Based Object Detection via Boundary Structure Segmentation,” *International Journal of Computer Vision*, vol. 99, no. 2, pp. 123–146, Sep. 2012. [Online]. Available: <http://link.springer.com/10.1007/s11263-012-0521-z>
- [105] H. Li, Q. Zhao, X. Li, and X. Zhang, “Object detection based on color and shape features for service robot in semi-structured indoor environment,” *International Journal of Intelligent Robotics and Applications*, vol. 3, no. 4, pp. 430–442, Dec. 2019. [Online]. Available: <http://link.springer.com/10.1007/s41315-019-00113-3>
- [106] Jiebo Luo and D. Crandall, “Color object detection using spatial-color joint probability functions,” *IEEE Transactions on Image Processing*, vol. 15, no. 6, pp. 1443–1453, Jun. 2006. [Online]. Available: <http://ieeexplore.ieee.org/document/1632198/>
- [107] N. Markuš, M. Frljak, I. S. Pandžić, J. Ahlberg, and R. Forchheimer, “Object Detection with Pixel Intensity Comparisons Organized in Decision Trees,” 2013, publisher: arXiv Version Number: 5. [Online]. Available: <https://arxiv.org/abs/1305.4537>
- [108] L. Neumann and J. Matas, “Scene Text Localization and Recognition with Oriented Stroke Detection,” in *2013 IEEE International Conference on Computer Vision*. Sydney, Australia: IEEE, Dec. 2013, pp. 97–104. [Online]. Available: <http://ieeexplore.ieee.org/document/6751121/>
- [109] D. Chen, G. Hua, F. Wen, and J. Sun, “Supervised Transformer Network for Efficient Face Detection,” in *Computer Vision – ECCV 2016*, B. Leibe, J. Matas, N. Sebe, and M. Welling, Eds. Cham: Springer International Publishing, 2016, vol. 9909, pp. 122–138, series Title: Lecture Notes in Computer Science. [Online]. Available: http://link.springer.com/10.1007/978-3-319-46454-1_8
- [110] X. Shi, S. Shan, M. Kan, S. Wu, and X. Chen, “Real-Time Rotation-Invariant Face Detection with Progressive Calibration Networks,” 2018, publisher: arXiv Version Number: 3. [Online]. Available: <https://arxiv.org/abs/1804.06039>
- [111] J. Ma, W. Shao, H. Ye, L. Wang, H. Wang, Y. Zheng, and X. Xue, “Arbitrary-Oriented Scene Text Detection via Rotation Proposals,” *IEEE Transactions on Multimedia*, vol. 20, no. 11, pp. 3111–3122, Nov. 2018. [Online]. Available: <https://ieeexplore.ieee.org/document/8323240/>
- [112] S. Ren, K. He, R. Girshick, and J. Sun, “Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks,” *IEEE Transactions on Pattern Analysis*

- and Machine Intelligence*, vol. 39, no. 6, pp. 1137–1149, Jun. 2017. [Online]. Available: <http://ieeexplore.ieee.org/document/7485869/>
- [113] J. Mao, M. Niu, H. Bai, X. Liang, H. Xu, and C. Xu, “Pyramid R-CNN: Towards Better Performance and Adaptability for 3D Object Detection,” 2021, publisher: arXiv Version Number: 1. [Online]. Available: <https://arxiv.org/abs/2109.02499>
- [114] H. Zhang, H. Chang, B. Ma, N. Wang, and X. Chen, “Dynamic R-CNN: Towards High Quality Object Detection via Dynamic Training,” 2020, publisher: arXiv Version Number: 2. [Online]. Available: <https://arxiv.org/abs/2004.06002>
- [115] X. Xie, G. Cheng, J. Wang, X. Yao, and J. Han, “Oriented R-CNN for Object Detection,” in *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*. Montreal, QC, Canada: IEEE, Oct. 2021, pp. 3500–3509. [Online]. Available: <https://ieeexplore.ieee.org/document/9710901/>
- [116] J. Pang, K. Chen, J. Shi, H. Feng, W. Ouyang, and D. Lin, “Libra R-CNN: Towards Balanced Learning for Object Detection,” in *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. Long Beach, CA, USA: IEEE, Jun. 2019, pp. 821–830. [Online]. Available: <https://ieeexplore.ieee.org/document/8953703/>
- [117] A. Keivani, J.-R. Tapamo, and F. Ghayoor, “Motion-based moving object detection and tracking using automatic K-means,” in *2017 IEEE AFRICON*. Cape Town: IEEE, Sep. 2017, pp. 32–37. [Online]. Available: <http://ieeexplore.ieee.org/document/8095451/>
- [118] C. Zhan, X. Duan, S. Xu, Z. Song, and M. Luo, “An Improved Moving Object Detection Algorithm Based on Frame Difference and Edge Detection,” in *Fourth International Conference on Image and Graphics (ICIG 2007)*. Sichuan: IEEE, Aug. 2007, pp. 519–523. [Online]. Available: <https://ieeexplore.ieee.org/document/4297140/>
- [119] T. Athanasiadis, P. Mylonas, Y. Avrithis, and S. Kollias, “Semantic Image Segmentation and Object Labeling,” *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 17, no. 3, pp. 298–312, Mar. 2007. [Online]. Available: <http://ieeexplore.ieee.org/document/4118230/>
- [120] J. C. Rangel, J. Martínez-Gómez, C. Romero-González, I. García-Varea, and M. Cazorla, “Semi-supervised 3D object recognition through CNN labeling,” *Applied Soft Computing*, vol. 65, pp. 603–613, Apr. 2018. [Online]. Available: <https://linkinghub.elsevier.com/retrieve/pii/S1568494618300553>

- [121] B. Bhanu and J. Han, “Kinematic-based human motion analysis in infrared sequences,” in *Sixth IEEE Workshop on Applications of Computer Vision, 2002. (WACV 2002). Proceedings*. Orlando, FL, USA: IEEE Comput. Soc, 2002, pp. 208–212. [Online]. Available: <http://ieeexplore.ieee.org/document/1182183/>
- [122] S. Eminoglu, M. Isikhan, N. Bayhan, M. A. Gulden, O. S. Incedere, S. T. Soyer, S. Kocak, C. Yalcin, M. C. B. Ustundag, O. Turan, U. Eksi, and T. Akin, “A 1280×1024 - $15\mu\text{m}$ CTIA ROIC for SWIR FPAs,” in *Infrared Technology and Applications XLI*, B. F. Andresen, G. F. Fulop, C. M. Hanson, and P. R. Norton, Eds., vol. 9451, International Society for Optics and Photonics. SPIE, 2015, pp. 218 – 230. [Online]. Available: <https://doi.org/10.1117/12.2179537>
- [123] K. Murari, R. Etienne-Cummings, N. V. Thakor, and G. Cauwenberghs, “A cmos in-pixel ctia high-sensitivity fluorescence imager,” *IEEE Transactions on Biomedical Circuits and Systems*, vol. 5, no. 5, pp. 449–458, 2011.
- [124] A. Berkovich, A. Castro, M. Islam, F. Choa, G. Barrows, and P. Abshire, “Dark current reduction by an adaptive ctia photocircuit for room temperature swir sensing,” in *2017 IEEE International Symposium on Circuits and Systems (ISCAS)*, 2017, pp. 1–4.
- [125] Y. Zhai and R. Ding, “Design of a ROIC with high dynamic range for LWIR FPAs,” in *Infrared, Millimeter-Wave, and Terahertz Technologies III*, C. Zhang, X.-C. Zhang, and M. Tani, Eds., vol. 9275, International Society for Optics and Photonics. SPIE, 2014, pp. 160 – 167. [Online]. Available: <https://doi.org/10.1117/12.2071838>
- [126] E. D. Borniol, F. Guellec, P. Castelein, A. Rouvié, J.-A. Robo, and J.-L. Reverchon, “High-performance 640×512 pixel hybrid InGaAs image sensor for night vision,” in *Infrared Technology and Applications XXXVIII*, B. F. Andresen, G. F. Fulop, and P. R. Norton, Eds., vol. 8353, International Society for Optics and Photonics. SPIE, 2012, pp. 88 – 95. [Online]. Available: <https://doi.org/10.1117/12.921086>
- [127] D. A. V. Blerkom, “Analysis and simulation of CTIA-based pixel reset noise,” in *Infrared Technology and Applications XXXVII*, B. F. Andresen, G. F. Fulop, and P. R. Norton, Eds., vol. 8012, International Society for Optics and Photonics. SPIE, 2011, pp. 159 – 168. [Online]. Available: <https://doi.org/10.1117/12.886958>
- [128] P. Zarkesh-Ha, “An intelligent readout circuit for infrared multispectral remote sensing,” in *2014 IEEE 57th International Midwest Symposium on Circuits and Systems (MWS-CAS)*. IEEE, 2014, pp. 153–156.

- [129] A. Berkovich, M. Lecca, L. Gasparini, P. A. Abshire, and M. Gottardi, “A 30 μ w 30 fps 110 \times 110 pixels vision sensor embedding local binary patterns,” *IEEE Journal of Solid-State Circuits*, vol. 50, no. 9, pp. 2138–2148, 2015.
- [130] G. Hermosilla, J. Ruiz-del-Solar, R. Verschae, and M. Correa, “A comparative study of thermal face recognition methods in unconstrained environments,” *Pattern Recognition*, vol. 45, no. 0, p. 2445 – 2459, Jul 2012. [Online]. Available: <http://www.sciencedirect.com/science/article/pii/S0031320312000179>
- [131] S. Z. Li, R. Chu, S. Liao, and L. Zhang, “Illumination invariant face recognition using near-infrared images,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 29, no. 4, pp. 627–639, 2007.
- [132] R. Shoja Ghiass, “Face Recognition Using Infrared Vision,” Doctoral Thesis, Université Laval, Québec, Canada, 2014. [Online]. Available: <http://www.qirt.org/liens/FMTV.htm>
- [133] R. Shoja Ghiass, H. Bendada, and X. Maldague, “Université Laval Face Motion and Time-Lapse Video Database (UL-FMTV),” in *Proceedings of the 2018 International Conference on Quantitative InfraRed Thermography*. QIRT Council, 2018. [Online]. Available: <http://qirt.org/archives/qirt2018/papers/051.pdf>
- [134] A. Georghiadis, P. Belhumeur, and D. Kriegman, “From few to many: Illumination cone models for face recognition under variable lighting and pose,” *IEEE Trans. Pattern Anal. Mach. Intelligence*, vol. 23, no. 6, pp. 643–660, 2001.
- [135] Z. Sun and Y. Yu, “Fast approximation for sparse coding with applications to object recognition,” *Sensors*, vol. 21, no. 4, 2021. [Online]. Available: <https://www.mdpi.com/1424-8220/21/4/1442>
- [136] C. Meola, Ed., *Infrared Thermography Recent Advances and Future Trends*. BENTHAM SCIENCE PUBLISHERS, Jul. 2012. [Online]. Available: <http://www.eurekaselect.com/101682/volume/1>
- [137] S. Bench and R. Mieziako, “Terravic Research Infrared Database,” 2005. [Online]. Available: <http://vcipl-okstate.org/pbvs/bench/Data/05/download.html>
- [138] R. Padilla, S. L. Netto, and E. A. B. da Silva, “A survey on performance metrics for object-detection algorithms,” in *2020 International Conference on Systems, Signals and Image Processing (IWSSIP)*, 2020, pp. 237–242.