“CAMERA ELECTRONICS AND IMAGE ENHANCEMENT SOFTWARE FOR INFRARED DETECTOR ARRAYS”

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF APPLIED SCIENCES
OF
MIDDLE EAST TECHNICHAL UNIVERSITY

BY

ALPER KÜÇÜKKÖMÜRLER

IN PARTIAL FULFILLMENT OF THE REQUIREMENT
FOR
THE DEGREE OF MASTER OF SCIENCE
IN
ELECTRICAL AND ELECTRONICS ENGINEERING

FEBRUARY, 2012

Approval of the thesis:

CAMERA ELECTRONICS AND IMAGE ENHANCEMENT SOFTWARE FOR INFRARED DETECTOR ARRAYS

Submitted by **ALPER KÜÇÜKKÖMÜRLER** in partial fulfillment of the requirements for the degree of Master of Science in Electrical and Electronics Engineering, Middle East Technical University by,

Prof. Dr. Canan Özgen _____
Dean, Graduate School of **Natural and Applied Sciences**

Prof. Dr. İsmet Erkmen _____
Head of Department, **Electrical and Electronics Eng. Dept.**

Prof. Dr. Tayfun Akın _____
Supervisor, **Electrical and Electronics Eng. Dept., METU**

Examining Committee Members

Prof. Dr. Gözde Bozdağı Akar _____
Electrical and Electronics Eng. Dept., METU

Prof. Dr. Tayfun Akın _____
Electrical and Electronics Eng. Dept., METU

Assoc. Prof. Dr. Haluk Külah _____
Electrical and Electronics Eng. Dept., METU

Assoc. Prof. Dr. İlkay Ulusoy _____
Electrical and Electronics Eng. Dept., METU

Dr. Selim Eminoğlu _____
Mikro-Tasarım San. Ve Tic. Ltd. Şti.

Date: 09.02.2012

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name, Last Name: Alper Küçükkömürler

Signature :

ABSTRACT

CAMERA ELECTRONICS AND IMAGE ENHANCEMENT SOFTWARE FOR INFRARED DETECTOR ARRAYS

Küçükkömürler, Alper
M. Sc., Department of Electrical and Electronics Engineering
Supervisor: Prof. Dr. Tayfun Akın

February 2012, 108 pages

This thesis aims to design and develop camera electronics and image enhancement software for infrared detector arrays. It first discusses the camera electronics suitable for infrared detector arrays, then it concentrates on image enhancement software that are implemented including defective pixel correction, contrast enhancement, noise reduction and pseudo coloring. After that, testing and results of the implemented algorithms were presented.

Camera electronics and circuit operation frequency are selected considering the available standard programmable devices and the output rate of the detector readout circuitry. The target device for implementation of algorithms was Xilinx Spartan – 3 XC3S1500 which is used in the camera tests at METU-MEMS Research and Applications Center. Considering the real time operation, the target clocking frequency for operation of the circuitry was selected as 2MHz. Image enhancement algorithms primarily aim to be implemented for 320 x 240 resolution detectors, however with parametric

implementation, they aim to support other resolutions, including 160 x 120 and 640 x 512. In addition, all implementations aim to be modular and reusable.

Various different approaches are used for image enhancement software: (i) defective pixel correction is achieved by using a selective median filtering approach, (ii) contrast enhancement is achieved by employing contrast stretching and histogram based methods, and (iii) noise reduction is achieved by implementing a spatial filter. In addition to these, four types of pseudo coloring methods were applied and tested.

Test results show that defective pixel correction algorithm operates at 20.0 MHz, with 0.0 x 10^{-3} RMS error from its MATLAB prototype, and contrast enhancement algorithms are able to operate at 3.3 MHz, with an average of 545.0 x 10^{-3} RMS error. Spatial filtering for noise reduction operates at 20.0 MHz, with a 2.6 x 10^{-3} RMS. Pseudo-coloring 125.0 MHz, with a 0.0 x 10^{-3} RMS deviation from its MATLAB prototype

Keywords: IR Imaging, Image Processing, Digital Design.

ÖZ

KIZILÖTESİ DETEKTÖR DİZİNLERİ İÇİN KAMERA ELEKTRONİĞİ VE GÖRÜNTÜ İYİLEŞTİRME YAZILIMI

Küçükkömürler, Alper
Yüksek Lisans, Elektrik Elektronik Mühendisliği Bölümü
Tez Yöneticisi : Prof. Dr. Tayfun Akın

Şubat 2012, 108 sayfa

Bu tez kızılötesi detektör dizinleri için kamera elektroniği ve görüntü işleme yazılımı tasarlamayı hedefler. Tezde ilk olarak kızılötesi detektör dizinlerine uygun kamera elektroniği tartışılmış ve daha sonra etkisiz piksel düzeltme, zıtlık iyileştirme, gürültü azaltma ve sözde-renklendirme algoritmalarını içeren görüntü iyileştirme yazılımı üzerine odaklanılmıştır. Daha sonra bu algoritmaların test edilmesi ve test sonuçları sunulmuştur.

Kamera elektroniği ve devrelerin çalışma frekansı standart programlanabilir devreler ve okuma devresinin çıktı hızı gözönünde bulundurularak belirlenmiştir. Gerçek zamanlı çalışabilmesi için bütün devrelerin en az bu frekansta çalışabilmesi gerekir. Algoritmaların gerçekleneceği cihaz olarak, ODTÜ-MEMS Araştırma Merkezi'nde testlerde kullanılmakta olan Xilinx Spartan – 3 XC3S1500 cihazı seçilmiştir. Gerçek zamanlı çalışma gereksinimleri gözönünde bulundurularak hedef saat frekansı olarak 2 MHz seçilmiştir. Görüntü iyileştirme algoritmaları birincil olarak 320 x 240 çözünürlükteki detektör dizinleri ile çalışabilmeyi hedeflemektedir, ancak parametrik

yapısı sayesinde 160 x 120 ve 640 x 512 de dahil olmak üzere diğer çozünürlüklerde de çalışmayı amaçlamaktadır. Bunlara ek olarak bütün tasarımlar modüler ve yeniden kullanılabilir olmayı hedeflemektedir.

Görüntü iyileştirme için çeşitli yöntemler kullanılmıştır: (i) etkisiz piksel düzeltme, seçici medyan filtreleme yöntemi ile (ii) zıtlık iyileştirme, zıtlık genişletme ve histogram tabanlı yöntemlerle, (iii) gürültü azaltma, mekansal filtreleme yöntemiyle yapılmıştır. Son olarak dört çeşit sözde renklendirme yöntemi uygulanmış ve denenmiştir.

Test sonuçları etkisiz pixel düzeltme yöntemi 20.0 MHz frekansında çalıştığı ve MATLAB prototipinden 0.0 x 10^{-3} RMS farkla, ve zıtlık iyileştirme yöntemlerinin 3.3 MHz frekansında çalıştığı ve ortalama 545.0 x 10^{-3} RMS farkla sonuç verdiğini göstermiştir. Mekansal filtreleme ile gürültü azaltma 20.0 MHz frekansında çalışarak 2.6 x 10^{-3} RMS farkla sonuç verirken sözde renklendirme göntemi 125.0 MHZ frekansında çalışarak MATLAB prototipinden 0.0 x 10^{-3} RMS farkla sonuç üretmiştir.

Anahtar Kelimeler: Kızılötesi Detektörler, Görüntü İşleme, Sayısal Tasarım.

To My Parents

ACKNOWLEDGEMENTS

I would like to express my sincere gratitude and appreciation towards my advisor Prof. Dr. Tayfun Akın; Dr Selim Eminoğlu, and Assoc. Prof. Dr. İlkay Ulusoy for their facilitation and guidance throughout my study and thesis work in METU. I also would like to thank Dr. Murat Tepegöz, Fırat Tankut, Ceren Tüfekçi, Alperen Toprak, and Dinçay Akçören for their assistance with the works related with this thesis.

I also would like to thank my fellow members of METU MEMS –VLSI Research Group for their friendship and for providing a supporting environment to work in.

Last but not least, I would like to thank my family for their continuous support and encouragement through all my life.

TABLE OF CONTENTS

LIST OF FIGURES

FIGURES

LIST OF TABLES

TABLES

LIST OF ABBREVIATIONS

A/D :	Analog to Digital
CDF :	Cumulative Density Function
CS :	Chip Select
D/A :	Digital to Analog
EPROM :	Electrically Programmable Read Only Memory
FIFO :	First in First out
FPA:	Focal Plane Array
FPGA :	Field Programmable Gate Array
GPU :	Graphics Processing Unit
IC :	Integrated Circuit
IR :	Infrared
MCU :	Micro Controller Unit
MSPS :	Mega Sample Per Second
PCB :	Printed Circuit Board
PDF :	Probability Density Function
PLD :	Programmable Logic Device
PLL :	Phase Locked Loop
RAM :	Random Access Memory
RGB :	Red, Green, Blue
RMS :	Root Mean Square
SCK :	SPI Clock
SDI :	Serial Data In
SDRAM :	Synchronous Dynamic Random Access Memory
SE :	Shift Enable
SMQT :	Successive Mean Quantization Transform
SPI :	Serial Peripheral Interface
USB :	Universal Serial Bus

CHAPTER 1

INTRODUCTION

Electromagnetic waves have a wide spectrum, from relatively low frequencies, that we call radio waves, to higher frequencies, that we call light, to even higher frequencies that are high energy particles. Human eyes can sense the waves with wavelengths of 400 nanometers to 760 nanometers, the longest being perceived as red and the shortest being perceived as violet colors. Light with shorter wavelengths than that of violet are called *ultraviolet* and ones with longer wavelengths than that of red are called *infrared* lights.

Infrared light has some significant properties. For example its refraction properties in some materials are different than visible light. Some objects that are visible or opaque in visible light might appear transparent or even invisible in infrared light. A large part of thermal radiation, especially around room temperature, falls into infrared spectrum. In other words the objects around these temperatures emit light in infrared spectrum.

These properties of infrared light give it unique uses for a variety of applications. For example dust particles that block the visibility in space observations are invisible to lights in infrared spectrum. Different refraction of infrared lights at different cloud types allows weather satellites to see through some clouds and distinguish between them.

Thermal radiation also makes infrared light useful for night vision since objects emit thermal light by themselves, no outside light sources are necessary for viewing them. In addition, thermal vision allows fault finding in circuits, detection of some health problems in humans, and automated target tracking in military applications.

For thermal vision, there is not only a need for proper IR image acquisition electronics, but also image processing algorithms for enhancing the captured image in order for the user to make best use of it. This thesis focuses on both developing camera electronics for proper IR image acquisition and implementing image processing algorithms for enhancement of these images.

Section 1.1 emphasizes the significance of flexible electronic structures and states why the FPGAs are central to both the camera electronics and image enhancement software. Section 1.2 provides more information on IR image acquisition and requirements of the camera electronics. Section 1.3 states and discusses the signal processing requirements of IR images, and what algorithms are used for these requirements. Finally, Section 1.4 provides a summary of the research objectives and a description of the organization of the thesis.

1.1 Significance of Flexible Electronic Structures

As the technology advances, the need for intermediary communication and signal processing hardware also increases. Since the communication and signal processing requirements of each circuit is different than one another, these hardware need to be flexible to address these needs.

In order to address these requirements, several approaches were used throughout the history of development of electronic circuits. Early devices in this category were *Programmable Logic Devices* (PLDs). These devices were designed in a way to allow user to fuse and burn the unnecessary connections between the arrays of gates, leaving the desired logic on de chip. Of course, these devices could only be programmed once and used for a single purpose only.

Around that time, another approach was to use memory units. Rather than their default usage, *Electrically Programmable Read-Only Memories* (EPROMs) were used as look up tables of logical operations. This way, complex logical operations could be

implemented by the user with a single device. Since these memory chips were already available, this was rather a more practical and cost sensitive approach.

Later, techniques for controlling the interconnections of gate arrays with electric fields were developed. These devices are called *Field Programmable Gate Arrays* (FPGAs). They are both flexible and reprogrammable, so they can be used and modified according to the needs of circuitry.

Today, modern FPGAs contain millions of gates, large memories, and complex arithmetical and logical blocks, and some FPGAs even contain several *Central Processing Units* (CPUs). They are an integral part of both the development cycle and life cycle of many circuits.

The ability of FPGAs to execute multiple processes simultaneously makes them a very useful platform for controlling digital circuits and signal processing operations. They can be specialized for the requirements of specific data transmission and signal processing requirements. Their weakness in the form of lower operation frequencies are compensated by ability to process data in a parallel manner. This is especially true for data driven applications where large chunks of data are manipulated by the device. Consequently, FPGA is preferred for the implementation of both camera electronics for the IR image acquisition and image enhancement due to the reasons mentioned above.

1.2 IR Image Acquisition

There are many methods of infrared image acquisition, and these can be grouped in two main categories: thermal detectors and photon detectors. In principle, thermal detectors use the heating properties of infrared radiation, and employ the changes in circuit characteristics such as resistance or diode potential in the target side affected by the infrared radiation. On the other hand, the photon detectors exploit the electron excitation due to the incoming photons in low bandgap semiconductors.

METU MEMS Research and Applications Center specializes on microbolometer type thermal infrared detectors. Microbolometers consist of a suspended structure for heat

isolation and an active material, which is a resistance or a diode, on it for sensing. When infrared radiation arrives at the suspended structure, it begins heating the structure. Being suspended prevents the structure from cooling off immediately and allows it to keep its temperature long enough to be read out. Figure 1.1 illustrates a suspended structure for IR detection.

As the temperature of the suspended structure increases, the temperature of the active material increases too. Due to this increase in the temperature, resistance or diode threshold of the active material will change depending on the type of the material. This will result in changing the output voltage of the sensor. This changing output voltage is later read out by a CMOS circuitry, and it's multiplexed to the output of the chip.

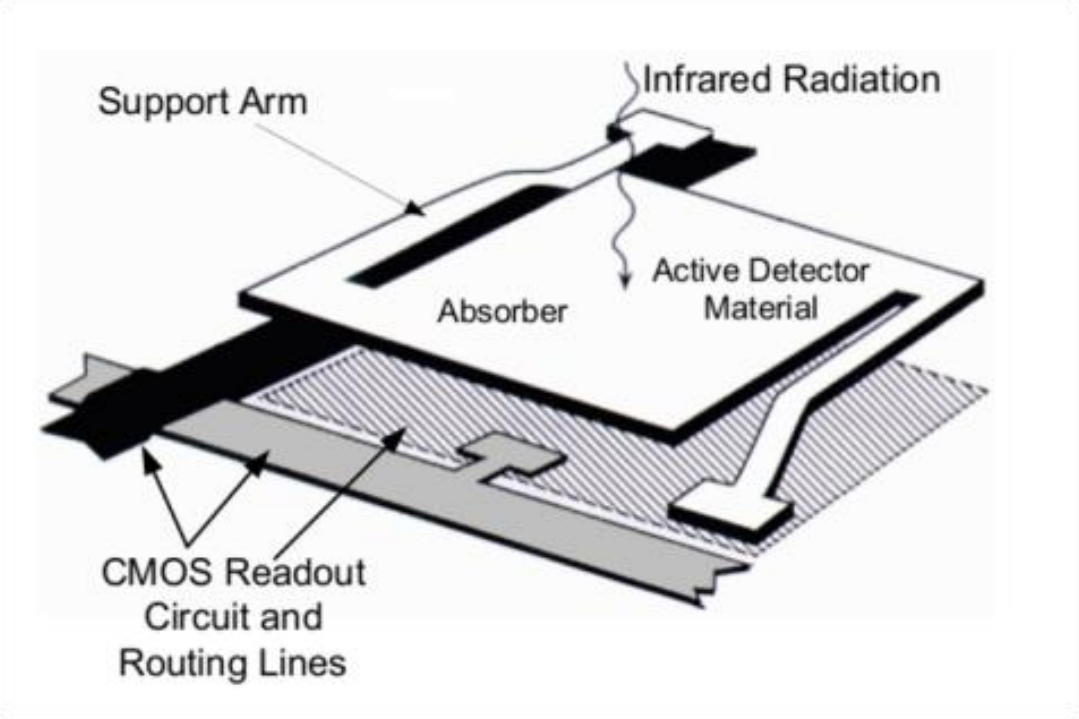


Figure 1. 1 A suspended structure for infrared radiation detection[1].

In METU MEMS Research and Applications Center, resistive type microbolometers are designed and produced for high performance imagery. Also more recently, diode type microbolometers, with relatively lower performance, but with a very easy fabrication, are developed for low cost imagery [2-8].

The infrared detector arrays are placed on some camera electronics that is controlled by an FPGA. The camera electronics must provide necessary power, proper biasing, and required digital signals in order for the detector array to operate properly. The output data of the array must be converted to a digital signal via A/D converters. This digital data then, needs to be collected in a buffer with additional metadata before being transferred to the host computer. Then, accompanying camera software must receive this data and construct an image.

1.3 Signal Processing Requirements of IR Images

After being captured, images from IR cameras are not always in the best format to be understood by humans and made meaning out from. The images may have noise introduced by sensors, readout electronics, or external circuitry. The output values of the IR camera may not be distributed on histogram evenly, thus they are not using it effectively. Some pixels in the image may be poorly responding or not responding at all. In addition, translating the output values of an imager to a gray scale might not be the best way to make meaning out of them.

For improving the condition due to defective pixels methods for defective pixel replacement including median filtering and selective median filtering can be employed. Noise on the image can be reduced using noise reduction methods such as spatial, temporal, or wavelet domain filtering. Contrast of the image can be enhanced either by contrast stretching or by histogram based methods like histogram equalization, plateau equalization, or adaptive histogram equalization. Furthermore pseudo-coloring can be employed in order for the user to more easily make meaning out of the images. This thesis focuses on the implementation of the methods on an FPGA.

1.4 Research Objectives and Thesis Organization

Two main objectives of this research are:

1. To develop proper IR image acquisition electronics for IR detector arrays with 320 x 240 resolution, but also include various resolutions such as: 160 x 120 and 640 x 512.
2. To implement image processing algorithms, including defective pixel correction, contrast enhancement, noise reduction, and pseudo-coloring that processes the output of the image acquisition electronics in real-time.

The imagers that are designed at the METU – MEMS Research and Applications Center require some biasing and some digital signals in order to operate properly, which need to be provided via an external circuitry. They scan the image in a horizontal order, producing their analog outputs as horizontal lines read one after another. This analog data requires to be converted to digital levels and organized in a manner so that it can be reorganized, before being transmitted to the host computer.

There are some requirements for an FPGA implementation that needs to be matched for image processing algorithms. In order for the system to produce images in real time, the incoming images need to be processed at the rate that the imaging system outputs it. Generally the imaging systems output the image data sequentially as one pixel at a time, and scanning lines or rows of the image depending on the design of the readout circuitry.

A recent readout circuitry designed in METU – MEMS Research and Applications Center [9] recommends a maximum clocking frequency of 2 MHz. This thesis uses that frequency as a design goal for the clocking frequency to be met by the implemented image enhancement methods.

The resolution of analog to digital conversion is another factor to be considered in the design. Although 14 bit A/D converters are used in most of the imagers in METU – MEMS Research and Applications Center, this thesis takes its design goal to implement generic methods that can be used in systems with different A/D conversion resolution.

The imaging devices in METU – MEMS Research and Applications Center use Spartan 3 family XC3S1500 FPGAs from Xilinx for timing of the circuits and data transfer. This is a large enough FPGA to fit some simple image enhancement methods in. Since this FPGA is already being used, this thesis takes its area as the area goal to be met.

As a resolution target 320 x 240 was selected, however designs are intended to be generic and be able to run for any resolution, including 160 x 120 and 640 x 512.

Since the FPGA is not a very large one, fixed point mathematics is chosen as a design criterion. All of the mathematical operations are implemented as fixed point operations. In the operations like division or square root which could result fractional numbers, any multiplication operations after them were referred back before these operations in order for minimum loss of data

As for any good design, the designs in this thesis aim to be reusable and modular. Parametric design is aimed for most of the modules, their bit lengths and buffer sizes are designed to be changed parametrically. Also designs need to be device generic; they need to be able to be used in various devices without much modification.

This thesis is organized as follows:

In Chapter 2 camera electronics is discussed in depth. In this chapter requirements of the imaging chip and how they are provided is presented. In addition to that, data transfer to the host computer and the accompanying camera software are presented.

Chapter 3 focuses on defective pixel correction methods, including median filtering and selective median filtering, and discusses the implementation of the latter.

Chapter 4 discusses contrast enhancement methods. In this chapter, methods such as contrast stretching, histogram equalization and its variations, successive mean quantization, and unsharp masking are presented. After that, implementations of four of these, including contrast stretching, histogram equalization, and two plateau equalization techniques are discussed.

Chapters 5 states the sources of the noise on the image and discusses how the noise can be reduced using spatial or temporal filters, or in the wavelet domain. After that it presents implementation of a spatial filter.

Chapter 6 is about pseudo-color methods. This chapter presents various palettes of pseudo-coloring, and how they are implemented on the FPGA.

Chapter 7 describes the test procedure by which the methods are tested, furthermore it provides the results and states the limiting factors.

Finally Chapter 8 discusses shortcoming of some algorithms and bottlenecks in the implementation, and presents a road map for future improvements.

CHAPTER 2

CAMERA ELECTRONICS

This chapter gives an overview of camera electronics. It describes how the chip is operated properly; data is captured, and transferred to the host computer, and it briefly mentions the camera software that interfaces with this camera electronics. Section 2.1 states the requirements of the camera electronics and describes the hardware. Section 2.2 describes how the IR detector arrays are operated properly. Section 2.3 describes the data acquisition and necessary components for it. Finally, Section 2.4 provides the details of the camera software.

2.1 IR Image Acquisition Electronics

IR detector arrays generally have a readout circuitry consisting of an analog part that detects changes of the voltage of the active material and amplifies it with as little noise as possible, and a digital part that controls multiplexers that output the analog signal of the corresponding pixels. This readout circuitry needs some analog voltages for biasing and some digital signals for synchronization, in addition to sufficient power in order to operate. [10] provides an in-depth discussion of the camera electronics, and how it relates to readout circuitry.

After the changes on the sensor are converted to voltage values via the readout circuitry, these analog voltages need to be converted to digital values in order for them to be transferred to the host computer. For this conversion an A/D converter and its FPGA interface is necessary.

But before the data is transferred to the host computer some additional information needs to be added in order to ensure the right value is displayed at the right place. With some additional metadata the system will be able to correctly reconstruct image correctly, furthermore it will be more robust in terms of data transmission losses.

As a final step, the captured image needs to be displayed on a screen through a camera software. Moreover, the software will enable the user to control the biasing of the circuitry, sampling the data and image processing operations.

The hardware that contains circuitry for powering and biasing the detector array and data acquisition, consists of several layers of printed circuit boards (PCBs). Figure 2.1 demonstrates the camera hardware. The topmost PCB contains the sensor and provides connections to it from the other layers. The following board contains the A/D converter and the two D/A converters; this board provides the necessary biasing for the imager chip and the A/D conversion of its output data. The following card holds the power circuitry for powering the previous boards. The final board provides the connection to the FPGA board.

For the transfer of the data to the host computer via USB XEM 3010 board from Opal Kelly was used. The board hosts a Spartan3 XC3S1500 FPGA, a MCU for USB data transfers. The FPGA is responsible for programming the chip and the D/A converters, clocking and sampling the A/D converter and transferring the image data to the host computer.

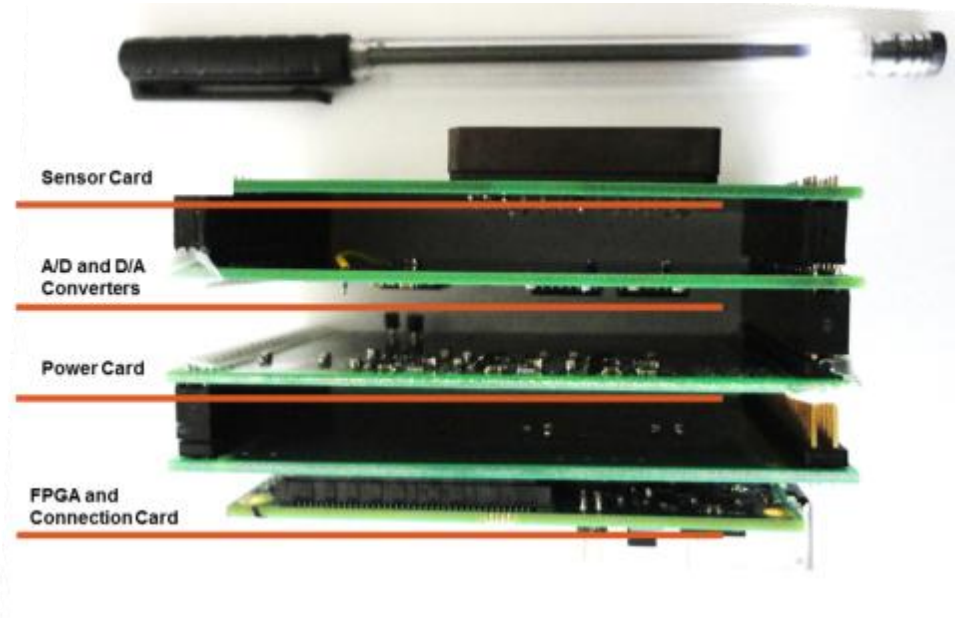


Figure 2. 1 The camera hardware.

2.2 Proper Operation of IR Detector Arrays

As for any circuitry IR detector arrays requires sufficient powering with as little noise or regulation possible. The power for the components in the circuitry is provided by several low dropout regulators. The regulator that was selected LT1762 by Linear Technology because of its low noise characteristics.

The analog circuitry requires some biasing both in terms of voltage and current in order to operate the active material and the amplification circuitry. The biasing that's required can be either implemented inside the readout chip, or be provided from an outside source. When implemented inside the chip, biasing can be controlled by means of digitally programming the chips corresponding digital circuitry. When it's not implemented inside the chip, it needs to be provided by external means, such as D/A converters. Generally, implementation strategy which is composed of a combination of the former and the latter is used.

For digitally programming the chip the SPI (Serial Peripheral Interface) protocol was used. This protocol employs three wires for the serial transfer of data: SDI, SE, and SCK. SDI (Serial Data In) is the wire that carries the data from the FPGA to the chip. SE (Shift Enable) acts as chip-select signal for marking the communication interval. Data transfers are synchronized with SCK (SPI Clock), at each SCK cycle if SE is high a bit of data is transferred via SDI from the FPGA to the IR detector array chip. A total of 16 bits are transmitted, first 8 bits carry the address data while the following 8 bits biasing information.

AD7304 from Analog Devices, which is an 8 bit, quad, rail-to-rail D/A converter, provides the external biasing. Similar to the imaging chip this device is programmed via the SPI protocol. Instead of SE this chip has a chip select (CS) signal that has the same function with the SE signal, when this signal is driven low, the 12 bit data transfer occurs. The first 2 bits are for power down of the device. The following 2 bits carry the address information that determines which one of the four channels will be programmed. The final 8 bits determine the value of the selected output.

Figure 2.2 demonstrates the State-machine diagram of the SPI programming module.

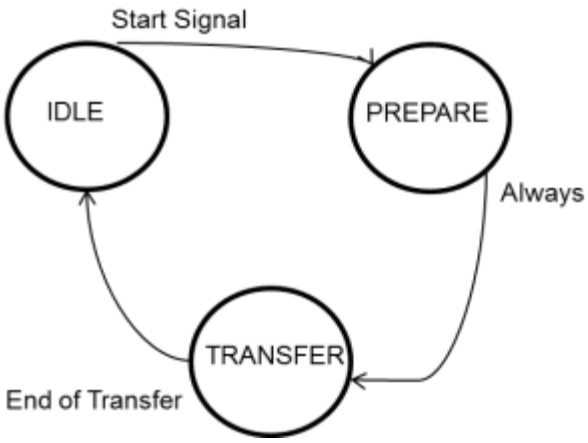


Figure 2. 2 The state machine for the SPI transfer.

In addition to analog biasing requirements, the imaging chips also require some digital signals in order to output the pixel data in the correct order. Some of the chips

require a clock signal together with the signals for synchronizing the line scanning and the frame scanning operations. There are also some imaging chips with integrated synchronizing circuitry that require only a main clock and generate required signals internally.

This synchronization can be implemented on an FPGA with two simple counters. The cooperation of these counters is summarized in Figure 2.3.

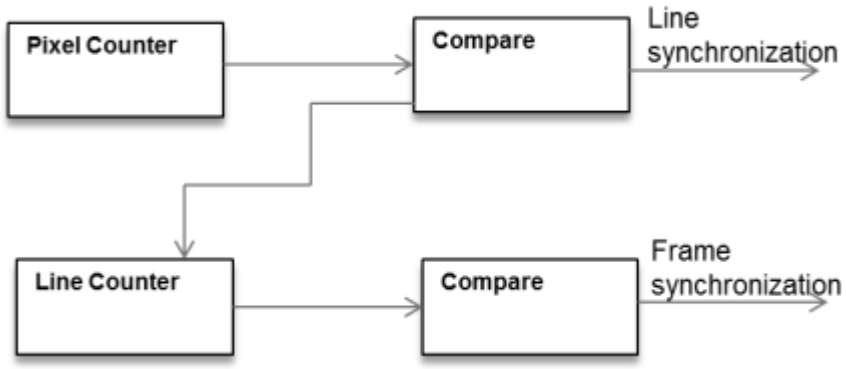


Figure 2. 3 Generation of the frame and line synchronization signals.

2.3 Data Acquisition

Imaging chips produce an output that consists of a series of sequentially multiplexed analog values, which are related to the illumination of the pixels. These values need to be converted to digital values via an A/D converter in order to be transferred to a host computer.

AD9240, a 14-bit 10 MSPS A/D converter from Analog Devices converts the analog data to digital values. The device samples the analog data at the positive edge of the clock and produces its output after a 3 clock cycle pipeline delay. For this reason, FPGA needs to generate a clock signal with an adjustable phase in order to have control over sampling.

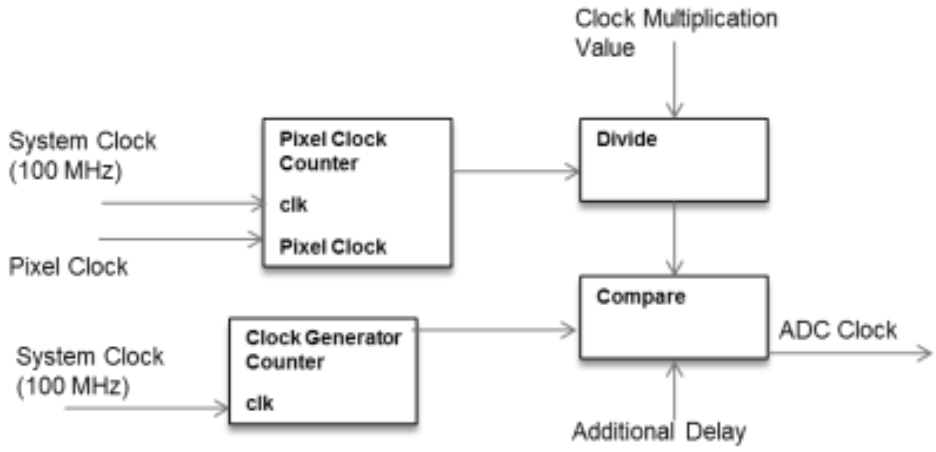


Figure 2. 4 Generation of the ADC Clock.

Figure 2.4 demonstrates the generation of ADC clock. The circuitry for generating ADC clock, counts the pixel clock as a function of system clock; this way it's ensured that the generated clock is at an exact phase with the pixel clock. ADC clock is generated as a function of the value of the second counter. The comparator compares the value of this second counter with the previously counted pixel clock and produces an ADC clock signal, and it adjusts the phase according to the *additional delay* input. For a slow pixel clock around 0.5 MHz, ADC clock generated with a 100 MHz system clock will have a 0.5% precision in terms of frequency.

After that the captured data needs to be sent to a host computer; however before this transmission, it's necessary to add some metadata to the bulk of captured pixel data for arranging them on the host computer. To achieve this, an additional tagging module is used. The module inserts tag data in the beginning of each frame and each line signaling which frame and which line the following data belongs to. A simple state machine which is implemented on the FPGA handles this operation. Generation of the metadata is illustrated in Figure 2.5.

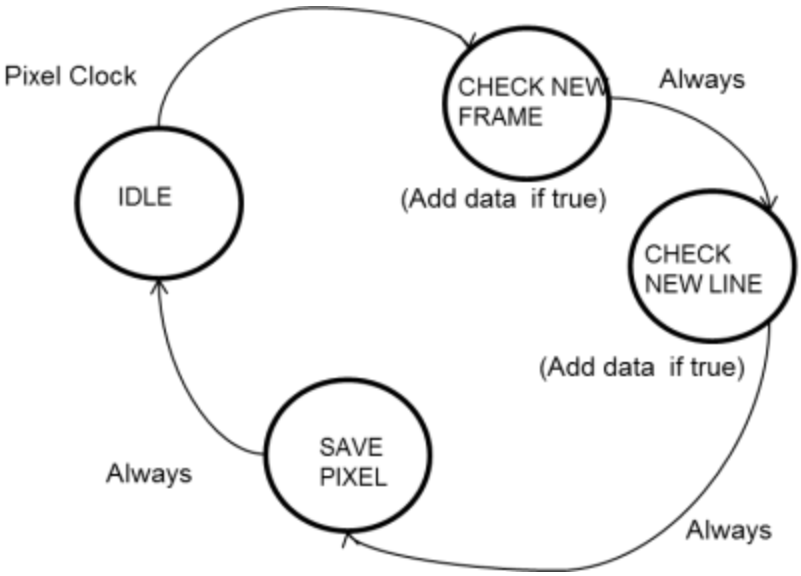


Figure 2. 5 State machine of the metadata generating circuitry.

Every time a pixel data is captured, the module checks if the data belongs to a new frame or new line. It adds a frame number and/or a line number if this is a new frame or a new line to the buffer that holds the pixel data.

2.4 Camera Software

The camera software is the host computer side complementary to the camera electronics. The user adjusts the biasing, and programs the camera circuitry using the camera software. In addition to that, the camera software displays the image captured by the sensor.

The software consists of three user interface windows: image display window, advanced settings window, and scope window.

Image display window is where the captured infrared image is displayed. This window communicates with the image transfer channel of the USB communication module. Each time the window is rendered, it checks the transfer buffer on the camera

for whether a complete image is captured or not. When there's a complete image in the transfer buffer this window transfers the image data into a memory using the metadata added by the FPGA, and then de-interlaces and does offset correction on it. Image drawing window is demonstrated in Figure 2.6.

Figure 2. 6 Image display window.

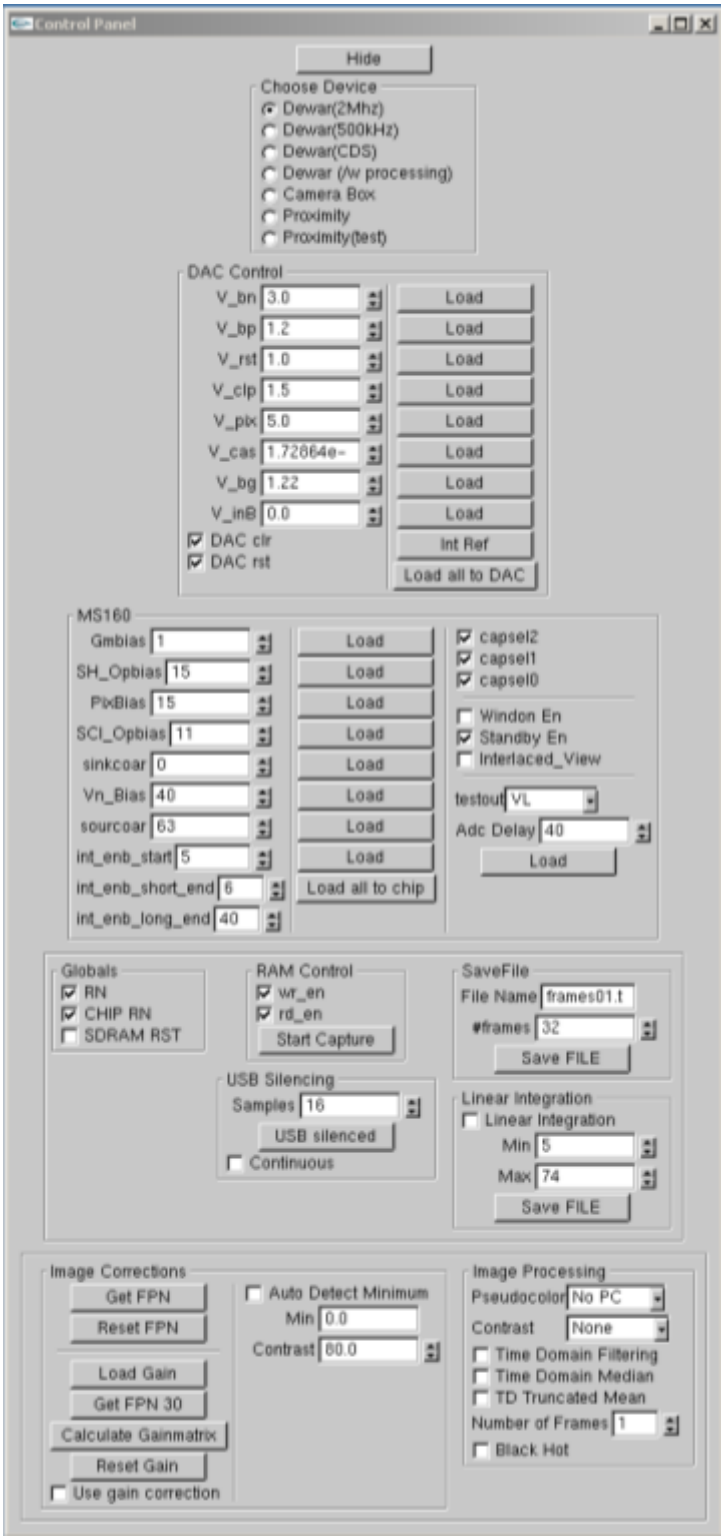


Figure 2. 7 Advanced settings window.

Figure 2.7 demonstrates the advanced settings window. The advanced settings window is responsible for programming the camera board components. It interfaces with transfer channel that interacts with the imaging chip, D/A, and A/D converters; and other general components of the FPGA. This window is, also the interface for some basic functions like saving data as a file, or activating or deactivating image processing functions.

The next component of the camera software is the scope window. This window acts as a virtual oscilloscope for observing the data output from the sensors. The window is implemented as a means to test the system without additional oscilloscopes, which would introduce additional load to the output buffers and change the observed data. The scope window is able to display the output of different lines, individual pixels or display the overall histogram of all sensor data. For each mode it can measure their mean values or noise levels instantly. Figure 2.8 demonstrates the scope window.

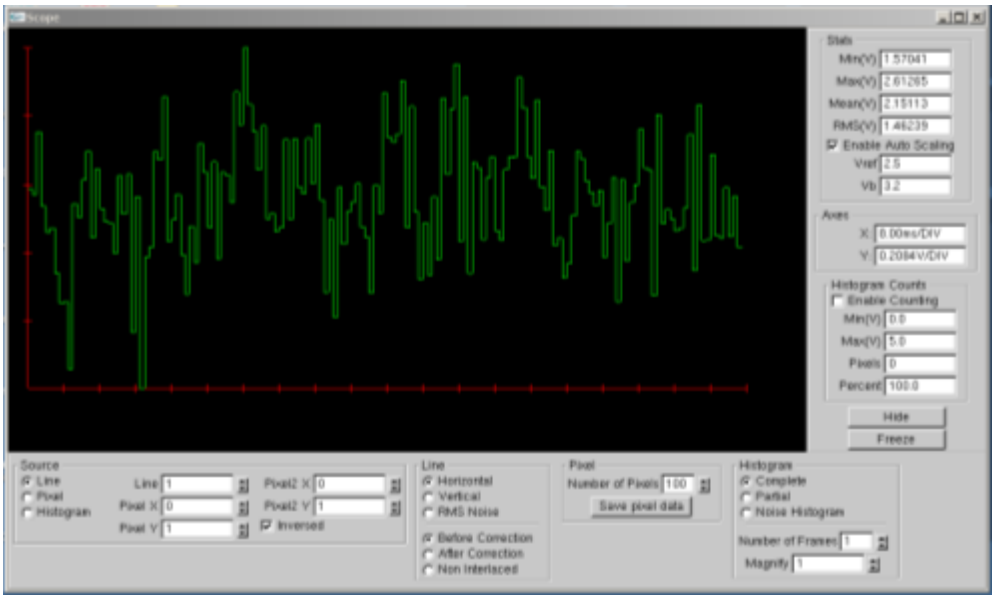


Figure 2. 8 Scope window.

CHAPTER 3

DEFECTIVE PIXEL CORRECTION

This chapter focuses on defective pixels and correction these pixels using median filtering and selective median filtering. Section 3.1 provides a discussion of defective pixels, and Section 3.2 describes median filtering. Then, Section 3.3 describes selective median filtering and states its advantages compared to median filtering, and finally Section 3.4 discusses the implementation of this method.

3.1 Defective Pixels

Uncooled Infrared detectors usually have an active material, a resistor or a diode depending on the type of detector; and a recess, created by a MEMS process, in order to prevent it from cooling. The voltage changes across the terminals of this active material due to infrared heating are measured by a readout circuitry. This readout circuitry outputs an analog value proportional to the change of the voltage across the terminals of the active material.

During the creation of these structures some imperfections may occur that change either electrical or thermal characteristics of these structures. Those changes cause them to respond to infrared radiation too weakly, too strongly or not respond at all. In addition to that, some defects may occur in the readout circuitry rendering the active material useless.

[11] discusses and characterizes the problem of defective pixels in depth. It classifies different kinds of defective pixels. Most important three of these classifications are: *stuck at low, stuck at high, abnormal sensitivity*. Stuck at low and high pixels provide an output that is at the low or high end of the voltage spectrum respectively irrespective of the incoming infrared stimulus. Similarly abnormal sensitivity pixels provide outputs with too much or too little sensitivity compared to other pixels. In this study all of these three kinds were observed.

3.2 Median Filtering

In basic image processing terms the noise introduced by defective pixels, where pixels give only black or white outputs, is called as salt and pepper noise since it resembles salt and pepper that stands on the image.

A simple and effective solution for decreasing salt and pepper noise is using a morphological filtering method. A moving average would help decrease the noise however this kind of noise would still be effective. Median filtering would yield better results since median of a series with extremes would better characterize it than its mean.

Median filtering can simply be applied by sorting the values and choosing the median value in an n by n kernel, for each pixel. Figure 3.1 demonstrates the result of median filtering.

3.3 Selective Median Filtering

The median filter is effective in situations like this. However, current infrared cameras can provide low resolution images at the range of, from 160 x 120 to 320 x 240 and 640 x 512 pixels. At these resolutions, detail lost by median filtering is too much compared to overall image size.

Figure 3.1 a) Raw image, b)Median filtered image.

Furthermore, median filtering might filter out important image details that are too small with respect to filter kernel size. For example a detail with size of a few pixels would be lost even with the smallest kernel.

An alternative to this method is to use median filtering in a selective manner, i.e., only to pixels that are known or estimated to be defective. After whether the pixel is effective or not is determined, the original pixel is replaced with the median of neighboring pixels.

For estimation of whether pixels are defective or not, another morphological filter, high pass filter namely, can be used. This way, defective pixels that are very different from their neighbors can be replaced with an estimated value, in this case median of the neighboring pixels. The defective pixels, on the other hand, which are very close to the mean of their neighboring pixels wouldn't be replaced.

This approach would preserve more details of original image compared to straightforward median filtering. However it would still cause a loss of important details when there are sudden spatial changes in the incoming image. Figure 3.2 provides comparison of the results of median filtering and selective median filtering.

Figure 3.3 and Figure 3.4 demonstrate the result of selective median filtering.

Since pixels defects are the result of imperfections in production, number of these pixels won't change during the usage of device. Knowing this, a better solution than this would be to measure the response of the pixels and keep a memory of effective and defective pixels. This kind of characterization of pixel responsivity is already being done in the production phase of the infrared camera chips.

After this initial characterization only the defective pixels can be replaced with an estimation whereas effective pixels remain unchanged. This would be a more robust technique compared to previous ones discussed. In addition none of the important details would be lost since all of the original values of effective sensors are kept unchanged. This can be implemented on a computer software, where memory isn't limited. For FPGA applications however, lack of memory may be a problem.

Figure 3. 2 a) Median filtered image, b) Selectively median filtered image.

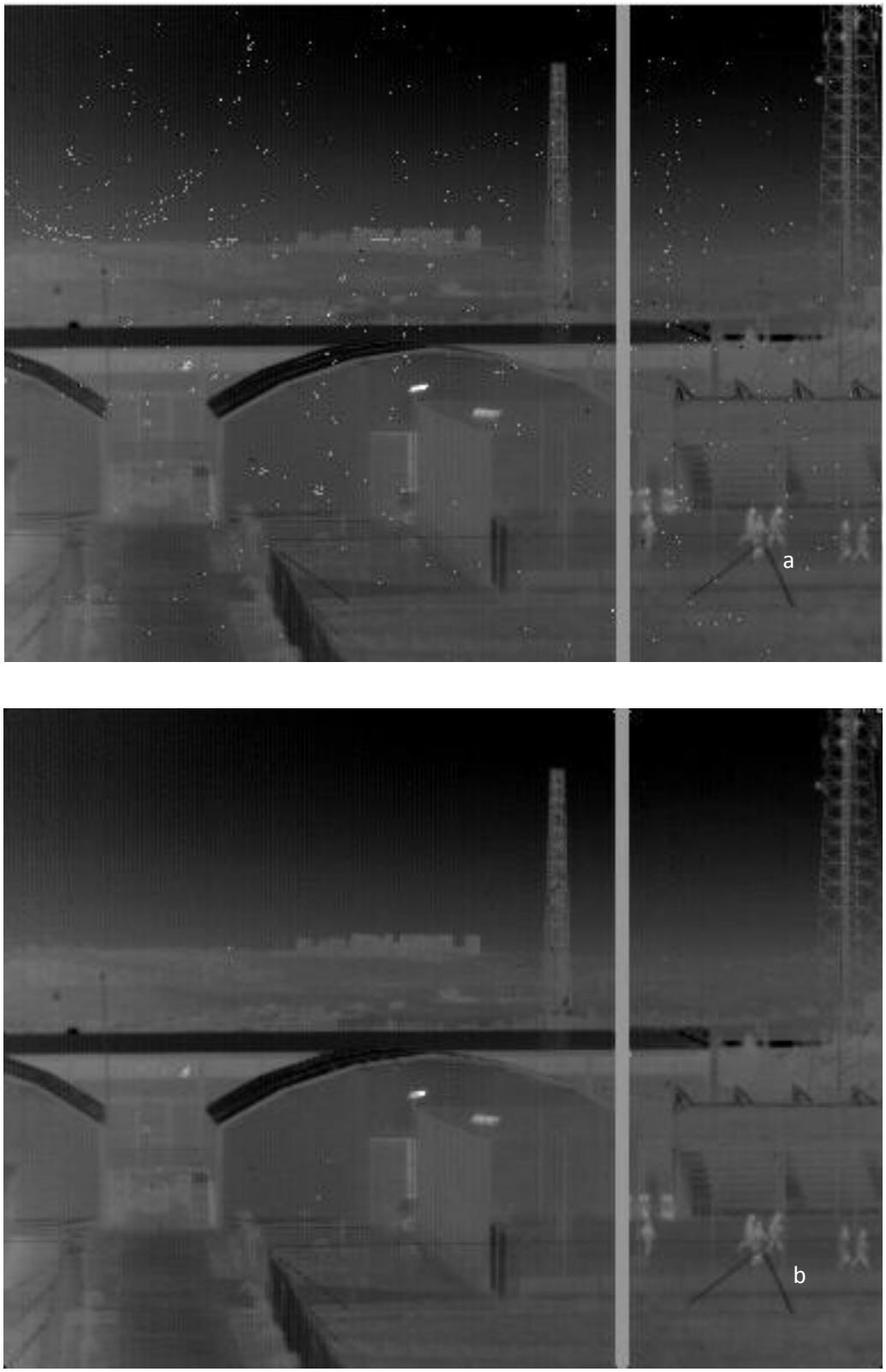


Figure 3. 3 a) Original image, b) Selectively median filtered image.

Figure 3. 4 a) Original image, b) Selectively median filtered image.

3.4 Implementation

The hardware implementation consists of a high pass filter and a median filter, working in parallel. Both of these filters require a kernel to work on. In software applications generally, an image stays stationary in the memory and a kernel moves on it to apply the filter. In hardware applications however, it's more compact and useful to keep the kernel stationary and pass the image data over it. [12] discusses a fast implementation for a convolution kernel.

The way a kernel is implemented is with an array of shift registers that holds the values of pixels that will be used for the calculation of the elements of the kernel and a series of fixed length FIFO buffers that hold the data of the remaining pixels in the line.

The pixels that enter the first row of shift registers form the first row of the kernel. Output of the first row is connected to the input of the first fixed length FIFO buffers. The length of this buffer must be shorter than the width of the image by the width of the shift registers. This buffer, together with the next row shift register, input of which is connected to this buffer, keep a number of pixel data which resembles a row of image. Following that, as many fixed length buffer – shift register pairs as necessary for the kernel is connected, one after another.

A high pass filter can simply be calculated as one minus a low pass filter. To keep the images' brightness level constant, generally low pass image filters have unity sum, and similarly high pass filters have zero sum. And to have zero sum the coefficients of the kernel are selected as floating point numbers, this is undesirable for this implementation since it will consume too much FPGA space. Rather than multiplying elements with floating point coefficients, they were multiplied with fixed point coefficients and then divided by the sum of these coefficients in order to get around this issue. Implementation of the high pass filter is demonstrated in Figure 3.5.

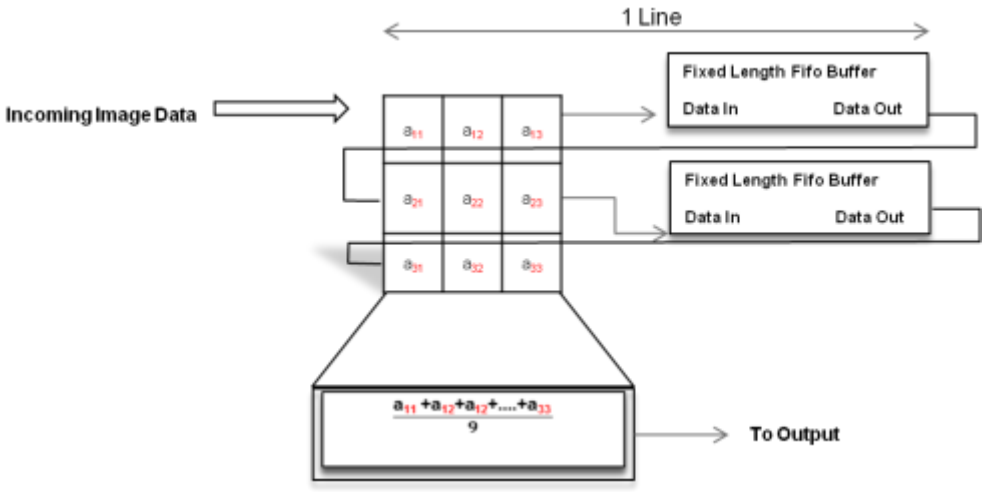


Figure 3.5 High pass filter implementation.

However, implementing division also has its drawbacks. Pipelined dividers also take up too much FPGA space. A better solution than implementing a pipelined divider is to divide the number with a power of two. Due to binary implementation of this operation it is enough to simply shift bitwise the value to be divided by the power to be divided by.

It was observed that high pass filters of size 3 by 3, elements of which are 1 except the center, yielding a total of 8 for low pass part of the kernel, provided satisfying results at the same time not requiring an additional divider. For this reasons this kernel was selected for implementation.

The high pass filter multiplies the elements in the kernel by one, and sums the elements around the center. It then divides this sum by eight and subtracts this value from the element in the center. This operation results in a high-pass image in which sudden spatial changes are represented with higher values.

The median filter similarly, consists of a kernel and a sorting network. The filter uses the same kernel that's already implemented for the high pass filter. The values in the kernel are fed into the sorting network where they will be sorted and the median is selected. Figure 3.6 demonstrates the implementation of the median filter.

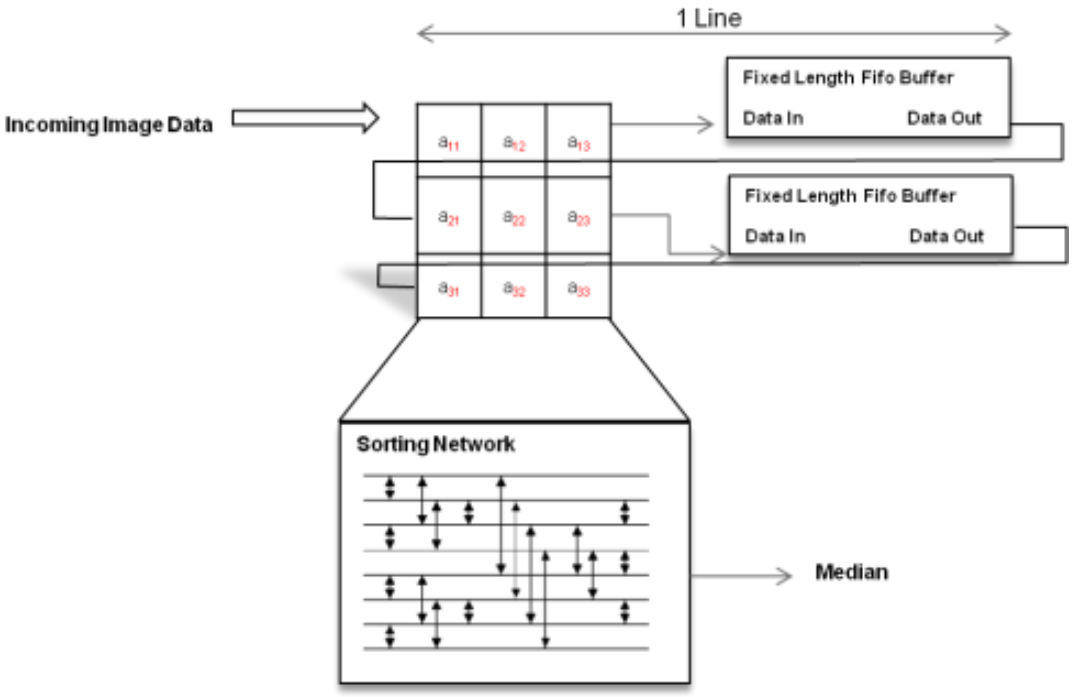


Figure 3.6 Median filter implementation.

The sorting network is made up of sub units called sorters. A sorter simply compares the numbers and swaps them if the number at the second input is larger than the number at the first input. Sorting is done in a manner that in software engineering terms, is called a *bubble sort*. Each element is compared to the nearby element and swapped if necessary. This operation is repeated as many times as the number of elements in the kernel. This repetition is implemented as pipelined layers so that in spite of the sorting delay it works with the same frequency that image data is input.

After the elements in the kernel are selected and sorted, the median can be found simply by selecting the middle item. The median value is selected as the output of the array, which is output from the sorting network, with the index that is half of the numbers in the kernel.

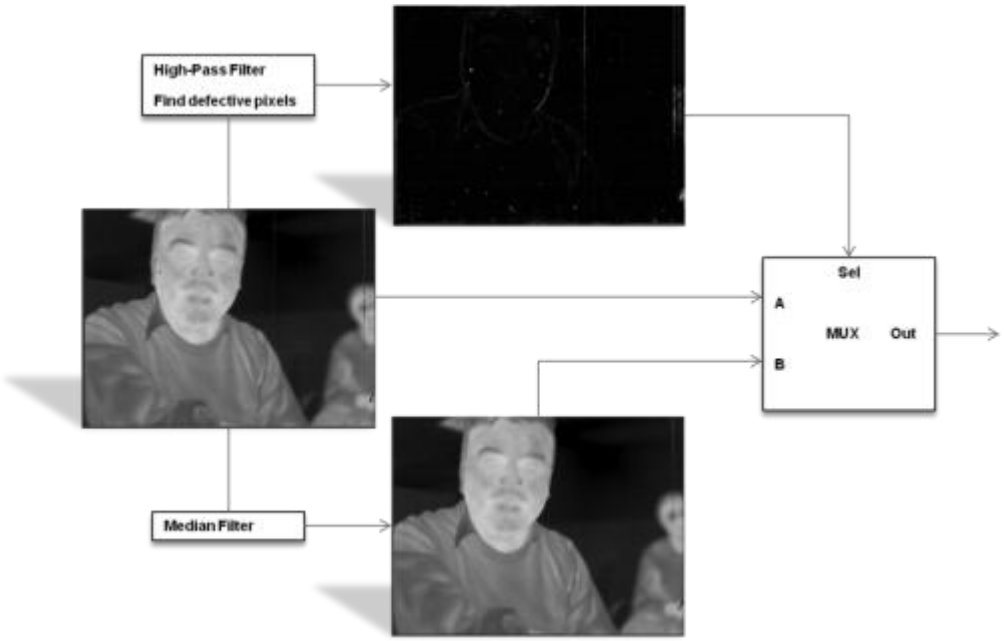


Figure 3.7 Selective application of median filter.

From this point on, a decision logic takes on the task of choosing whether to output the raw image data or the output of the median filter. The logic makes this decision according to the output of the high pass filter. The output value of the high pass filter is passed through a threshold, producing a binary result. This binary result indicates whether the related pixel is estimated to be a defective pixel or not. This operation is illustrated in Figure 3.7. The threshold can be either input from a top module, or can be a fixed number defined by a parameter. This choice is done during the synthesis time.

The median filter produces its result with a sorting delay. In order to select the corresponding pixels in the raw and the median filtered images, an additional delay is introduced on the raw image data in order to compensate for the delay of median filtering. The binary result generated by thresholding the output of the high pass filter controls a multiplexer, inputs of which are the median filtered image and the raw image. The output of this multiplexer forms the result of defective pixel correction.

CHAPTER 4

CONTRAST ENHANCEMENT

This chapter discusses contrast enhancement, algorithms for the enhancement of contrast and implementation of selected algorithms. Section 4.1 provides an overview of contrast enhancement. Section 4.2 presents and discusses commonly used contrast enhancement methods, including contrast stretching, histogram equalization, plateau equalization, contrast limited adaptive histogram equalization, successive mean quantization, and unsharp masking. Then, Section 4.3 presents implementations of contrast stretching, histogram equalization, and two plateau equalization methods, and then it discusses a contrast enhancement module capable of these four modes.

4.1 Contrast Enhancement

Generally, infrared images are captured in order to be seen and made meaning of by human users. An infrared image could be used to increase a driver's vision through the fog, or in order to detect relatively hotter objects in a field. Often, infrared images are used for seeing living creatures in the night. In cases like those, dynamic ranges of the original images, which is determined by the design of the sensor and the analog-to-digital circuitry around it, might not be always enough or in the best shape.

Infrared values that are read out by the camera are converted to gray levels relative to a reference and the dynamic range of readout circuitry. Often, images from the cameras employ less than the whole spectrum of the gray levels that the camera is

designed to have. More often than not, these images don't make the most efficient use of their histogram.

With some image processing, images can be modified to use the complete spectrum of all gray levels available, and to make better use of its histogram compared to unprocessed images. Image processing methods can also aid in improving the visibility of the background or foreground details, when necessary.

Both from the experience at the METU-MEMS Research and Applications Center and from the literature [13-18], some accurate assumptions for infrared images can be made. One accurate assumption is that, these images generally consist of a background, where there are some relatively cold objects and a foreground where there are hotter objects or agents. In more applications than not, the relatively hotter objects are the points of interest or more important than background details.

There are some well-known methods for image contrast enhancement such as contrast stretching and histogram equalization. And there are some less common methods that use image properties or geometric operations to enhance the contrast of the images.

Some examples in literature [13-18] use some modifications on the histogram equalization method using these assumptions to obtain better results. In addition to that there are some examples of using morphological operations [19] or image statistics and re-quantization [20] for enhancing the contrast of the image.

4.2 Contrast Enhancement Methods

4.2.1 Contrast Stretching

The simplest approach to use whole spectrum of the image is to stretch its values between the minimum and maximum of the displayable values. This is simply, done by transforming each pixel value by equation

$$I_{out} = \frac{(I_{in} - I_{min})}{(I_{max} - I_{min})} * I_{max,displayable} \qquad (4.1)$$

Where I_{min} and I_{max} are the minimum and maximum intensity values in the image, and $I_{max,displayable}$ is the maximum displayable value by the system. Figure 4.1 provides an example of contrast stretching.

4.2.2 Histogram Equalization

Histogram or probability density function (pdf) of a discrete valued image indicates the occurrence of an intensity level in the image and is denoted by:

$$pdf_x(i) = \frac{n_i}{n_{total}} \qquad (4.2)$$

Where n_i is the number of pixels in the i^{th} intensity level ($i = 1 \ldots . k$, k is the number of intensity levels) and n_{total} is the total number of the pixels in the image.

Figure 4.1 a) Raw image, b) Image enhanced with contrast stretching.

The aim of the histogram equalization method is to find a positive and monotonically increasing transform function that transforms the input image to an output image

$$x' = T(x) \quad (4.3)$$

Such that

$$pdf_{x'}(i) = \frac{\frac{n_{total}}{k}}{n_{total}} = \frac{1}{k} \qquad (4.4)$$

Also for both of these functions, their sum over all gray levels will be equal to 1.

$$\int_0^k pdf_x(i)\, di = \int_0^1 pdf_{x'}(j)\, dj = 1 \qquad (4.5)$$

Since the $pdf_{x'}(i)$ is constant relative to i and is equal to $\frac{1}{k}$

$$pdf_x(i)\, di = \frac{1}{k} dj \quad (4.6)$$

$$k\, pdf_x(i) = \frac{dj}{di} \qquad (4.7)$$

$$k(pdf_x(i)\, di) = dj \qquad (4.8)$$

Hence, the function that transforms x to x' will be

$$x'(i) = T(x(i)) = k \int_0^i pdf_x(\omega)\, d\omega \qquad (4.9)$$

For discrete valued images the integral will be equal to cumulative sum of the values of histogram up to ith level, which is also known as cumulative density function.

$$cdf_x(i) = \sum_{\omega=1}^{i} pdf_x(\omega) \quad (4.10)$$

Figure 4.2 demonstrates the result of histogram equalization.

Figure 4.2 a) Raw image, b) Image enhanced with histogram equalization.

4.2.2.1 Characteristic Histogram for the Infrared Images

As discussed earlier in this chapter, the assumption that infrared images consist of two main parts, the background and the foreground, namely, is accurate for most of the images. This situation can also be observed in a characteristic histogram of these images in general.

In a characteristic histogram generally, there is a peak at lower levels and, there are one or a few peaks at higher levels that, in most situations is the region of interest.

Most of the times, pixel numbers are larger under the peak that is associated with background, which means more pixels are related with background objects. With the histogram equalization method these levels are extended more than others. In other words background pixels are multiplied with larger gains.

In some cases this method might useful however; in some specific cases it's not very useful. Because it will decrease the visibility of foreground details since foreground pixels are multiplied with a relatively low gains. Furthermore it will increase the visibility of noise in the image.

To get around these some modifications of the image histogram can be used in histogram equalization. A simple method that can be employed is to use a point-wise linear histogram modification in which more important parts, such as foreground objects, of histogram are highlighted whereas less important parts, such as background, can be amplified less. This method would be especially useful when user adjusts the parts of the histogram that is the point of concern. A more generalized solution to get around would be to create plateaus in the places of large peaks. Those plateaus would reduce the effects of the peaks at the resulting image. Figure 4.3 demonstrates an IR image histogram and results of various histogram correction operations.

4.2.3 Plateau Equalization: Wang et. al.

Wang et. al. [14] proposed a straightforward algorithm and its hardware implementation for creating plateaus with adaptively calculating a threshold. Their proposed algorithm detects peaks first, and then sorts them; finally it chooses the median of the peaks as a threshold. The algorithm then reshapes the histogram by assigning the value of the threshold to the items that are larger than it. The thresholded parts in the histogram no longer causes disproportionately high gains allowing other parts to be better emphasized. Figure 4.4 provides an example of this operation.

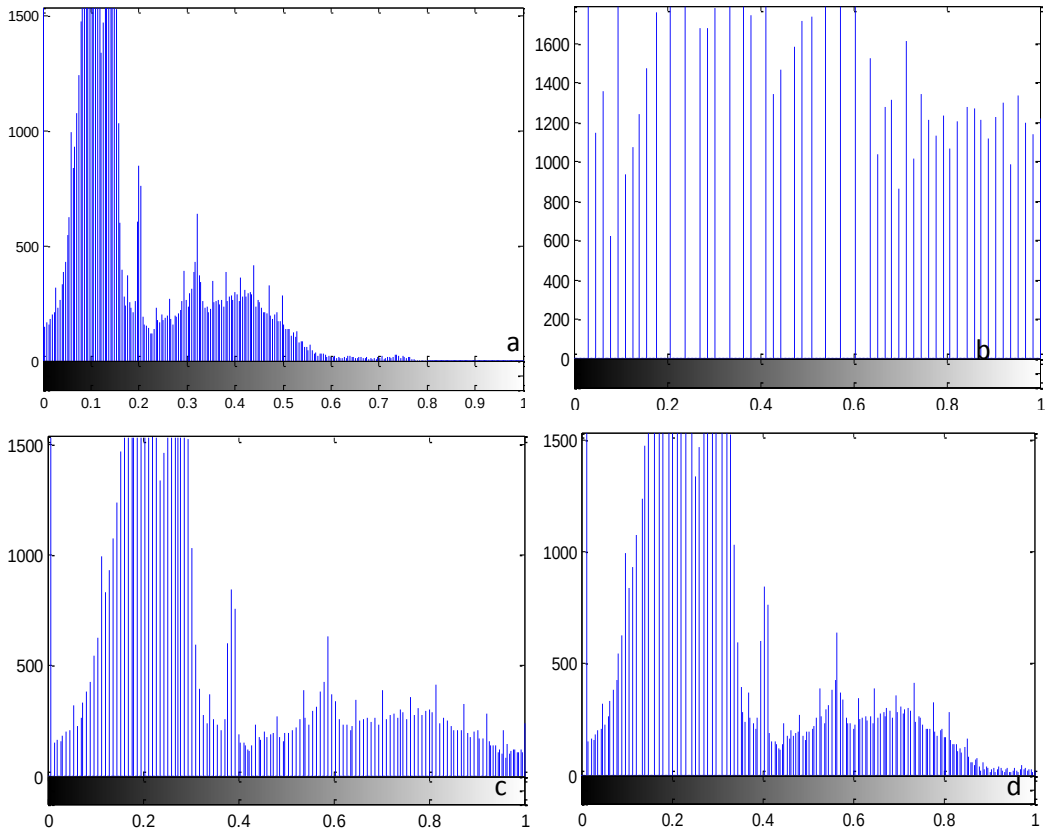


Figure 4.3 Histograms of a) raw image, b) image result of histogram equalization, c) image result of plateau method proposed by Wang et al., d) image result of plateau method proposed by Lai et al.

4.2.4 Plateau Equalization: Lai et. al.

Lai et al. [16] uses another technique for creating plateaus. Rather than thresholding the image histogram they scaled it. To do this they calculated the power of each value in the histogram divided by the maximum value of the histogram to a constant that is larger than zero and smaller than one.

$$pdf'(i) = \left[\frac{pdf(i)}{\max_{0<i<k}(pdf(i))}\right]^{0.31} \qquad (4.11)$$

Here, 0.31 is a value found empirically by a series of iterations that maximizes entropy in the resulting image.

By scaling the histogram, instead of thresholding, the algorithm keeps proportionality between values in the histogram, while at the same time reducing the effects of the larger peaks in the resulting image. Figure 4.5 provides an example of this operation.

Figure 4.4 a) Raw image, b) Image enhanced with plateau method proposed by Wang et al.

Figure 4.5 a) Raw image, b) Image enhanced with plateau method proposed by Lai et al.

4.2.5 Contrast Limited Adaptive Histogram Equalization (CLAHE)

Another approach that takes the properties of histogram into account is to adaptively enhance the contrast of the image by separating it into sub-regions is proposed by [17]. For each of these sub-regions then, a histogram is calculated and thresholded according to a desired contrast limit. In this method different from the plateau methods, the excess pixels that are clipped in thresholding phase are redistributed by taking into account the pixel distributions of neighboring pixels. The study in [18] also proposed and discussed an FPGA implementation for this method. Figure 4.7 provides an example of this operation.

4.2.6 Successive Mean Quantization Transform(SMQT)

The study in [19] suggested another method for contrast enhancement based on the image statistics. This method re-quantizes the image by levels determined by successively calculating the means of image and its sub regions.

The algorithm calculates the mean of the incoming image and assigns this value as the threshold for the most significant bit of re-quantization. After that, it divides the image into two regions as the pixels with values greater than the mean, and the ones with values lower than the mean. Following this step, the algorithm assigns these values as the threshold for the second most significant bit. This iteration continues as many times as the number of desired bits. The motivation behind this algorithm is to find a representation that is robust in terms of offset and gain applied to this image. The algorithm is illustrated in Figure 4.6. Figure 4.8 provides an example of this operation.

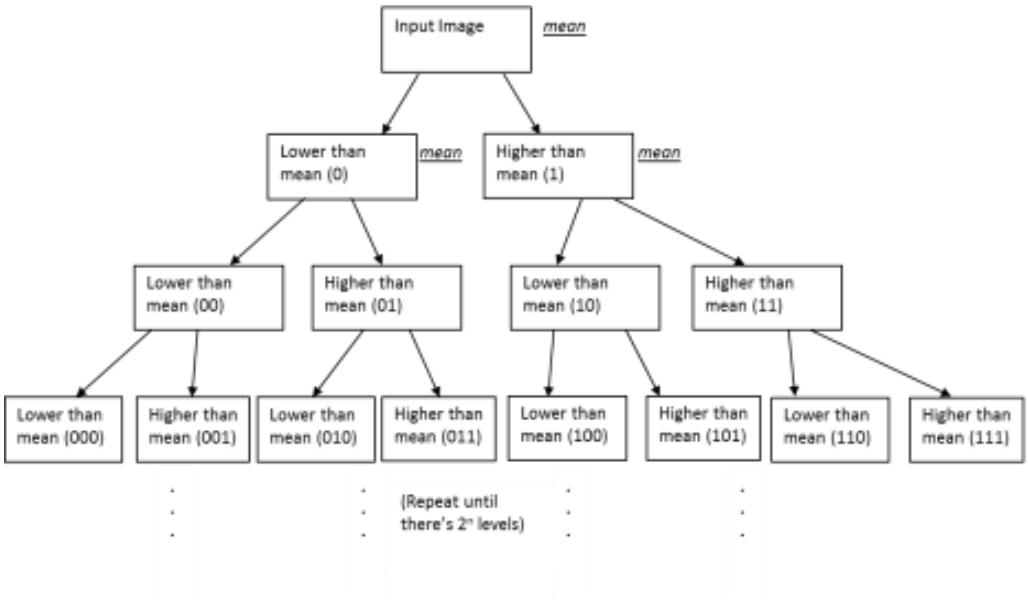


Figure 4.6 Successive Mean Quantization Transform (SMQT) algorithm.

This method can be especially useful in situations in which the temperature of both the environment and the observed object changes too much during the observation.

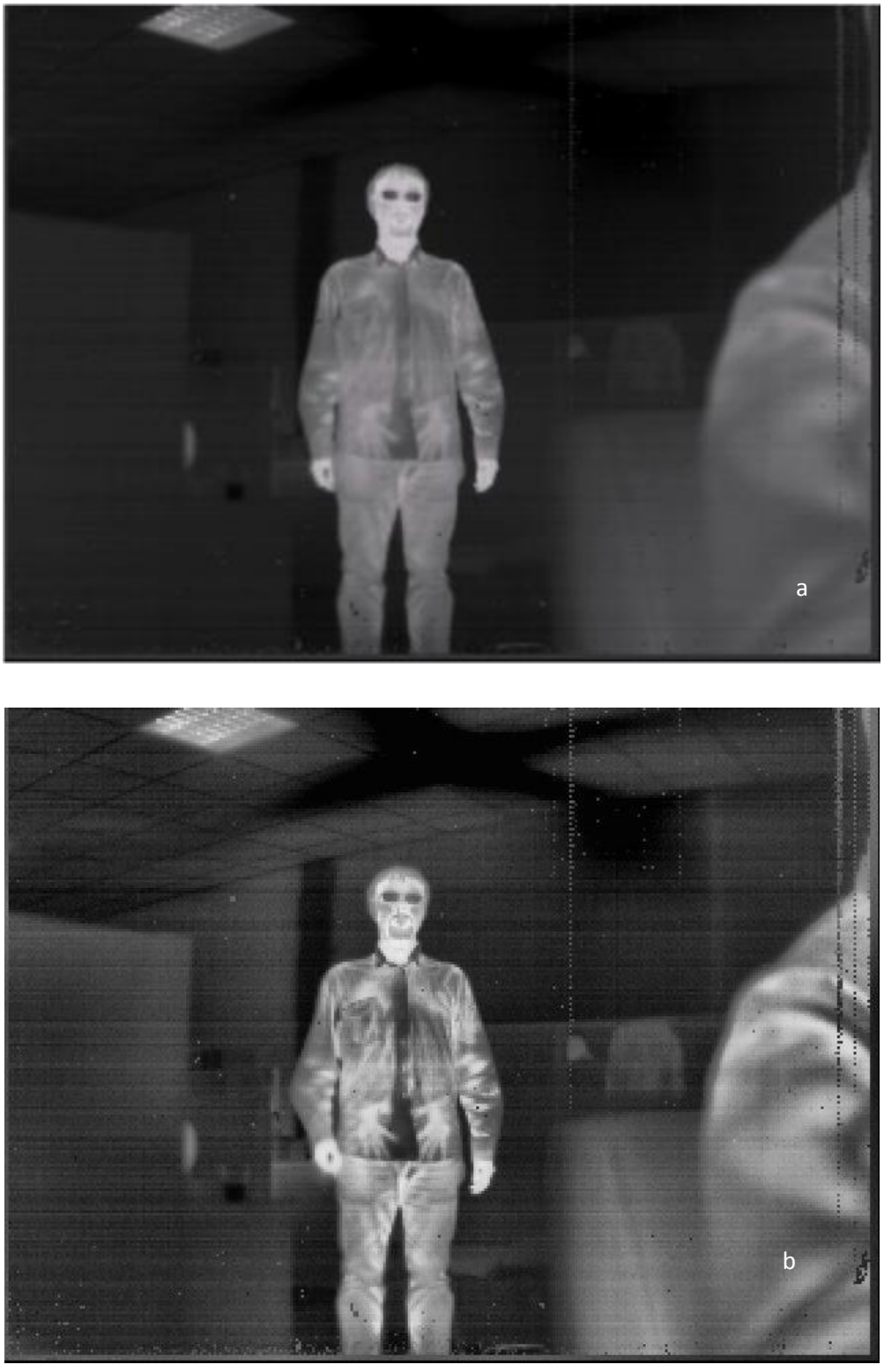


Figure 4.7 a) Original image, b)Image Enhanced with CLAHE.

Figure 4.8 a) Raw image, b) Image enhanced with SMQT.

4.2.7 Unsharp Masking

A widely used technique that is adopted from analog photography methods is unsharp masking. This method depends on the idea that when a low pass filtered image is subtracted from the original image, the resulting image will have amplified details.

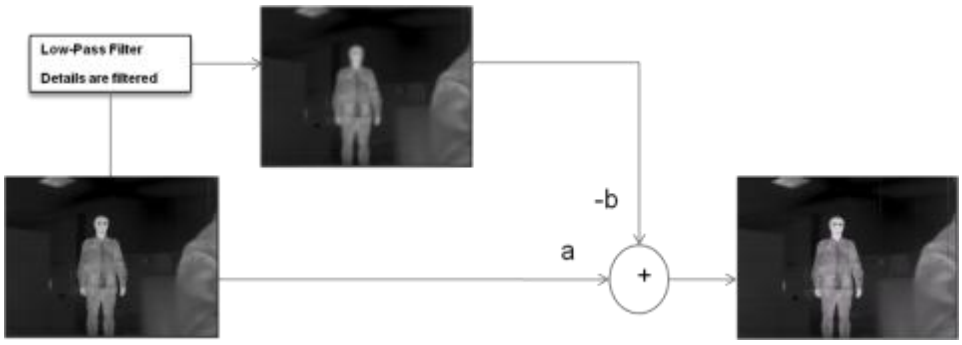


Figure 4.9 Unsharp masking.

Digital implementation of this filter is similar to the analog application. The input image is passed through a low pass filter, and then this low pass filtered image is subtracted from the original image with appropriate weights. This will result in an image with amplified details. This operation is illustrated in Figure 4.9. Figure 4.9 provides an example of this operation. Figure 4.11, Figure 4.12, Figure 4.13, and Figure 4.14 provide comparisons of the methods, which were discussed in this section.

Figure 4.10 a) Raw image, b) Image enhanced with Unsharp Masking.

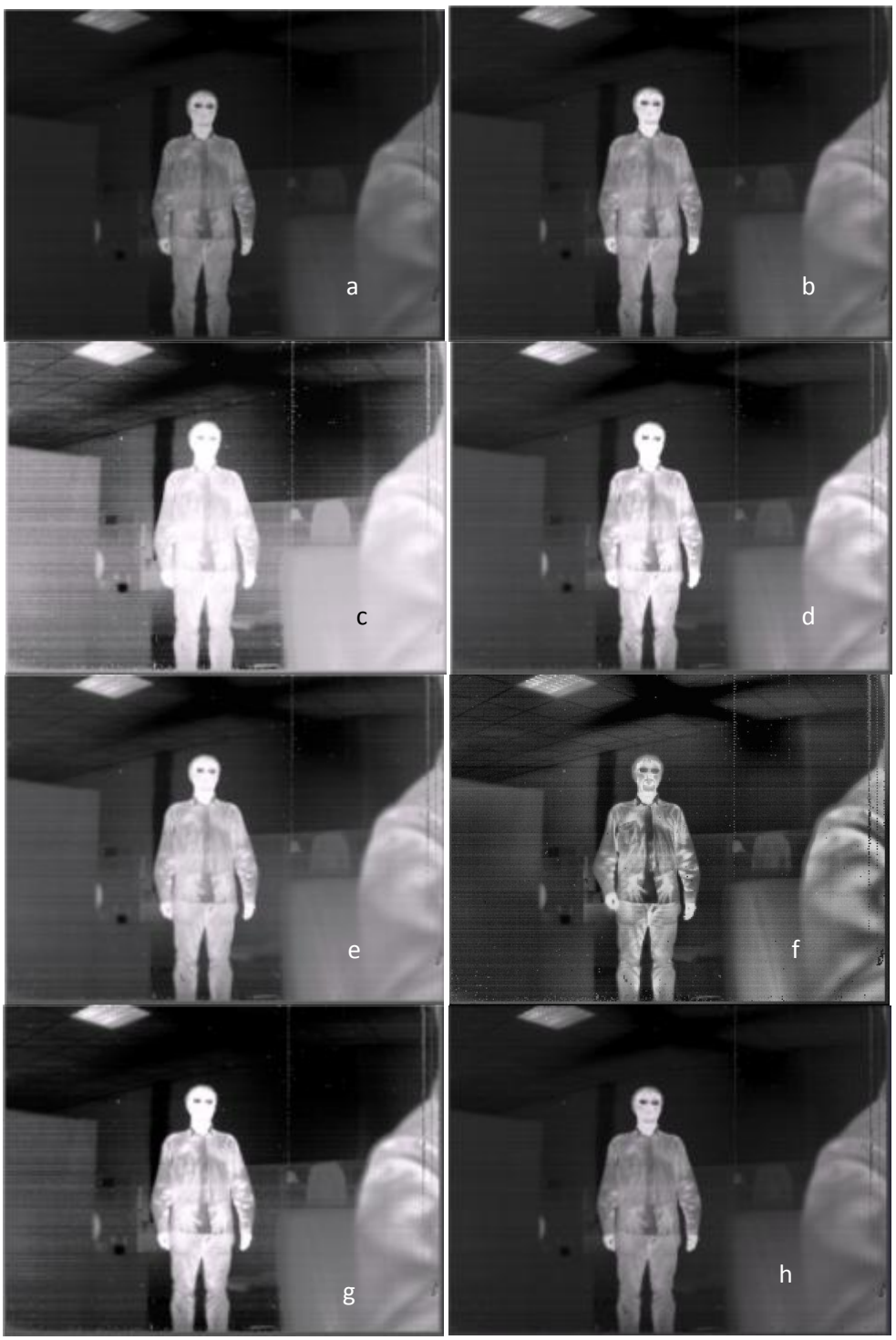


Figure 4.11 Results of contrast enhancement methods a) raw image, b) image result of contrast stretching, c) image result of histogram equalization, d) image result of plateau method proposed by Wang et al., e) image result of plateau method proposed by Lai et al., f) CLAHE, g) SMQT, and h) Unsharp masking.

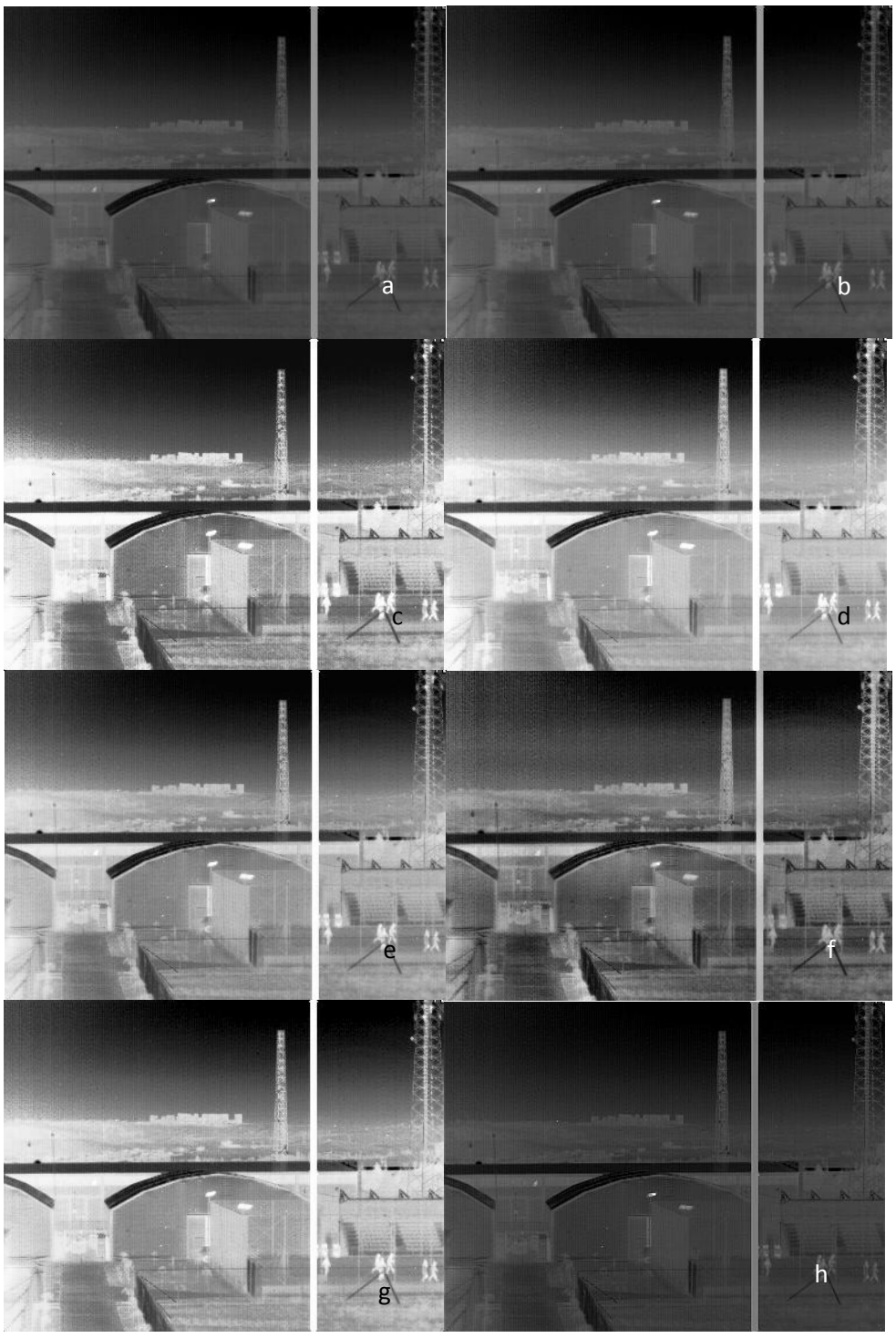


Figure 4.12 Results of contrast enhancement methods a) raw image, b) image result of contrast stretching c) image result of histogram equalization, d) image result of plateau method proposed by Wang et al., e) image result of plateau method proposed by Lai et al., f)CLAHE, g) SMQT, and h) Unsharp masking.

Figure 4.13 Results of contrast enhancement methods a) raw image, b) image result of contrast stretching c) image result of histogram equalization, d) image result of plateau method proposed by Wang et al., e) image result of plateau method proposed by Lai et al., f) CLAHE, g) SMQT, and h) Unsharp masking.

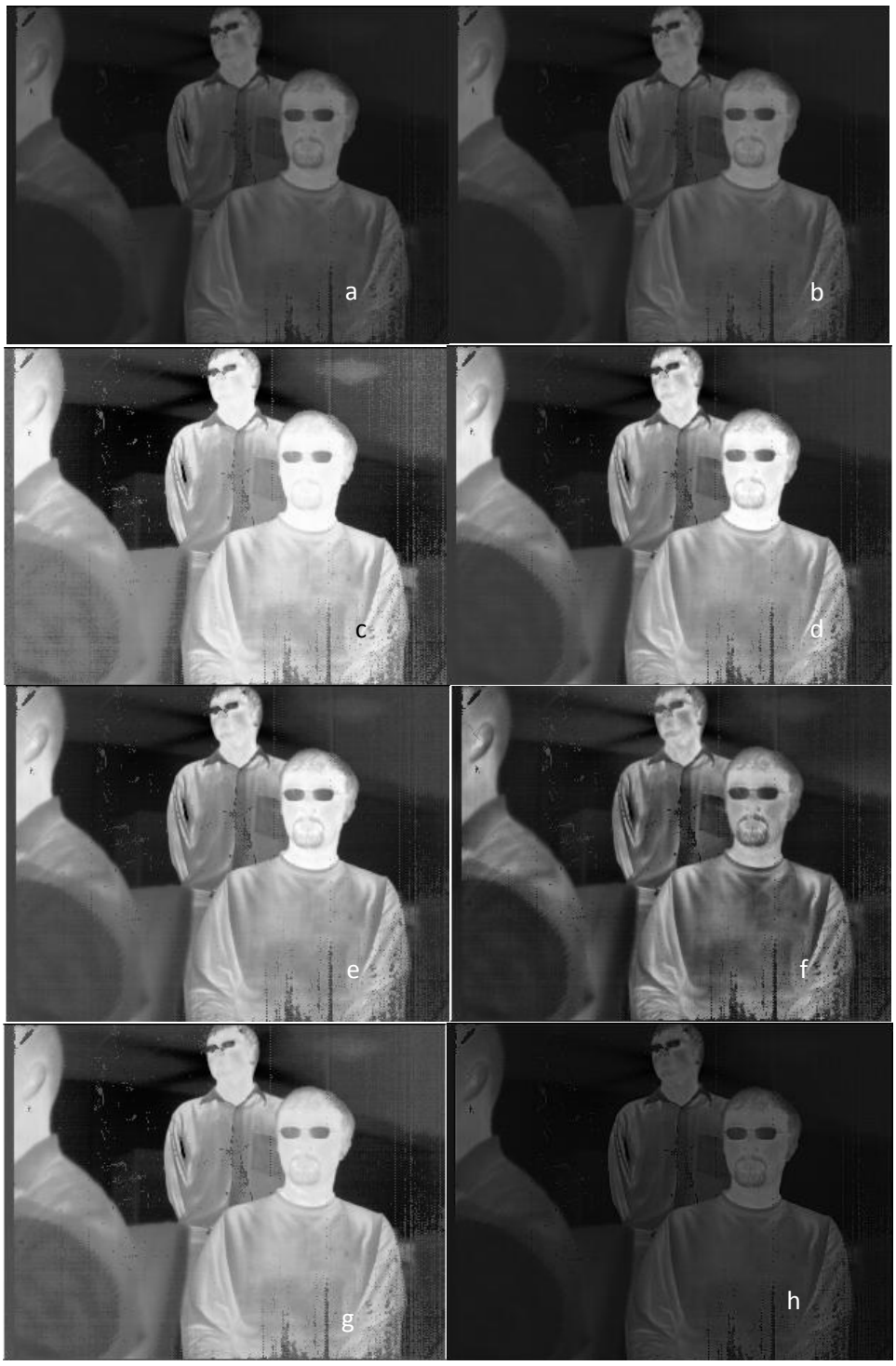


Figure 4.14 Results of contrast enhancement methods a) raw image, b) image result of contrast stretching c) image result of histogram equalization, d) image result of plateau method proposed by Wang et al., e) image result of plateau method proposed by Lai et al., f) CLAHE, g) SMQT, and h) Unsharp masking.

4.3 Implementation

Four methods are selected to be implemented for contrast enhancement because of their relatively better performances and suitability for FPGA implementation. These are: contrast stretching, histogram equalization, and histogram equalization with plateau methods proposed by Lai et al [16] and Wang et al [14]. These methods will have different uses in different situations.

The four methods that are implemented have some functions in common. For the optimum use of space they can combined into a single unit in which these common functions are shared between the methods. Because only one of the methods will be active in a given time, this sharing of functions can be facilitated.

The major bottlenecks, in terms of area, are memory and pipelined division. Using these modules in a shared manner would reduce the area that overall implementation consumes and allow the implementation of multiple contrast enhancement methods to be implemented on the same device.

4.3.1 Contrast Stretching Module

Contrast Stretching module consists of a maxima and a minima finder, a frame buffer, and the arithmetic operations. The maxima and minima finders are circuitry for detecting minima and maxima, the frame buffer is a memory unit and is controlled via a FIFO circuitry, and arithmetic operations circuitry is the circuitry for making the operation in the equation (4.1).

The maxima finder takes the value of incoming data, when the incoming data is larger than its current maximum value, and it's value is reset to zero every time a new frame signal is received so that the maxima belongs to the incoming frame and incoming frame only. Similarly minima finder takes the value of incoming data, when

the data is smaller than its current value, and it is preset to maximum value allowed for that ADC resolution range.

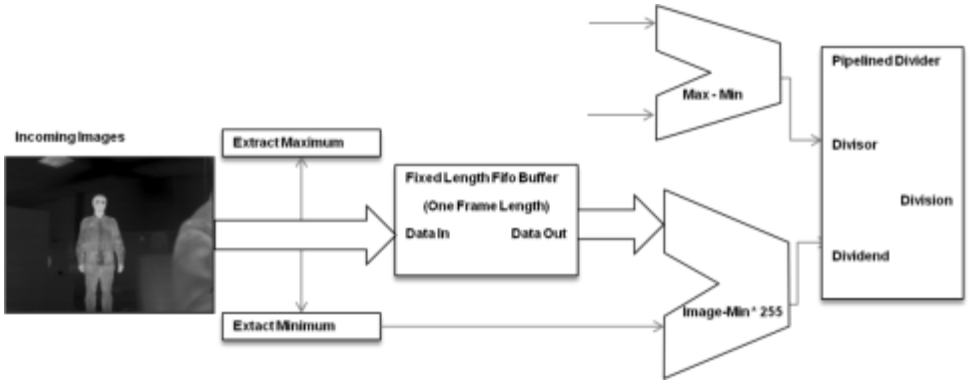


Figure 4.15 Contrast stretching implementation.

Frame buffer consists of a memory that is large enough to hold a complete frame and a circuitry to control it it in a First in - First out (FIFO) manner. This part of the circuitry is generically designed so that it can use the Block RAM of the FPGA or an external RAM, depending on the parameter input in the synthesis time. When using Block RAM, it generates the addressing and accesses the RAM accordingly during the synthesis. Conversely, in the external mode, it only generates addressing and outputs the addresses only.

These circuits that are described prepare the values in the equation (4.1). Since only fixed point operations are available, multiplication was implemented before division. After that, arithmetic operations are done on them. First, the minimum value is subtracted from each incoming value, then resulting number is multiplied with the absolute maximum value of the system; finally, result of this multiplication and the difference between the maximum and the minimum is sent to pipelined division module. Figure 4.15 illustrates contrast stretching operation.

This completes one cycle of processing, however when one frame is being processed, another frame is also being captured. There are two ways that the second frame can be handled. One of the ways is to hold a secondary FIFO buffer to keep the second frame, when the first one is being processed. The other way is to extract its

maximum and minimum values with a secondary detector set. The secondary detector set will extract the maximum and minimum of the second frame, while the data of the first set is being used for processing the first frame. For implementation, this second solution was used, for the reason that it would be more economical, in terms of FPGA space.

4.3.2 Histogram Equalization Module

Histogram equalization problem consists of two main parts. One of the parts is extracting the histogram, and the other part is calculating the lookup table and look up operation. The paper in [21] provides an in-depth discussion about histogram equalization with FPGAs.

Histogram Equalization module consists of two sub modules, namely *Image Statistics* sub-module and *Equalization* sub-module. In addition to those, the module includes a frame buffer, similar to one used in the Contrast Stretching module. *Image Statistics* sub-module extracts the histogram of the incoming frame, whereas *Equalization* sub-module calculates the cumulative density function from the histogram that *Image Statistics* sub-module extracted, and then uses it as a look-up table for the incoming image to be processed.

4.3.3 Image Statistics sub-module :

The purpose of the Image Statistics sub-module is to extract the histogram of the incoming frame. The module uses a Block RAM compound to extract and keep the histogram data. To do this, incoming image data is connected to the address port of the RAM block. Each time a pixel arrives, the data at this address is incremented. At the end of the frame, the data in the Block RAM would indicate the number of pixels in the gray level, at the corresponding address.

Xilinx FPGAs provide Block RAMs with *Read First* mode for simplifying operations like this. In *Read First* mode, an additional register is connected to the output of the RAM so that the data of the corresponding address is output at the first clock edge, and then at the second clock edge the data can be modified and re-written on the same address. This implementation uses this property to increment the data at the RAM associated with histogram. It should be also noted that this implementation requires twice the clock frequency of the incoming pixel data. This operation is illustrated in Figure 4.16.

After a complete frame is passed through Image Statistics sub-module, the module outputs the histogram data beginning from the lowest value, zero, and incrementing until the highest value, the number of gray levels. After that, data at all the addresses of Block RAM is cleared for the preparation for the next frame.

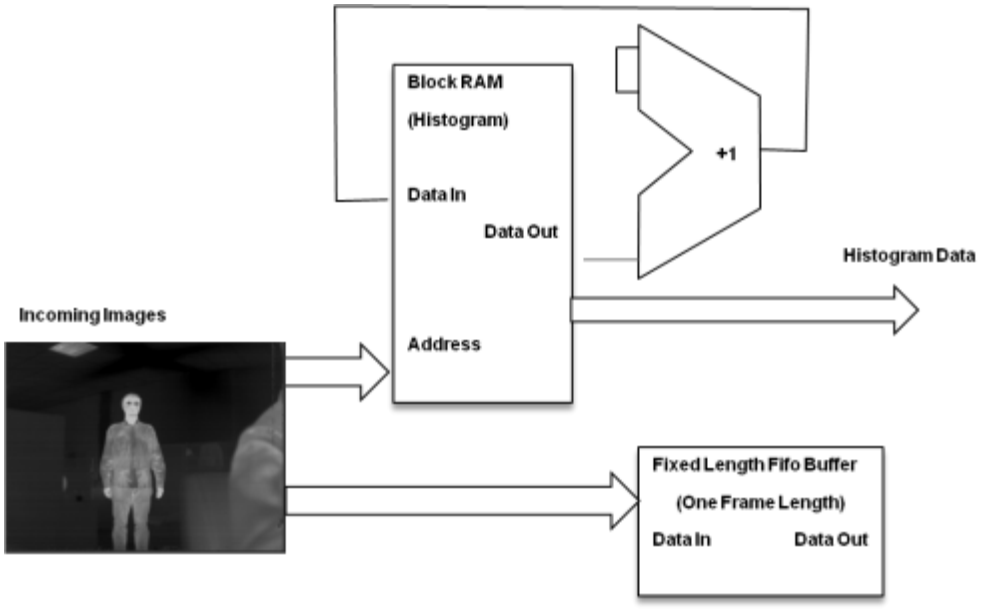


Figure 4.16 Histogram extraction circuitry.

For sequential frames, the situation discussed in the Contrast Stretching module applies for this module too. The contents of a RAM block needs to be output and cleared at the end of each frame. Again, to handle this delay an additional buffer or a

secondary RAM block can be used. The additional buffer must have a size that is large enough to keep image, while data is output and RAM is cleared. This requires twice the memory of Block RAM that's used for histogram, on the contrary adding another Block RAM for histogram, would use half of the memory that additional buffer would use. For that reason, the module includes two RAM blocks for histogram extraction. The timing of these blocks is summarized in Figure 4.17.

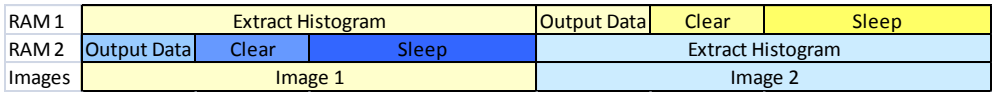


Figure 4.17 Timing of two RAM blocks in histogram extraction circuitry.

While the first RAM bock is busy extracting the statistics of one frame the second RAM block is at sleep state. At the end of a frame second RAM block begins to extract histogram of the next frame while first RAM block outputs the data and then clears itself. After the contents of the RAM are cleared, the RAM block then goes to sleep mode.

4.3.4 Equalization sub-module:

Equalization sub-module is responsible for calculating the cumulative sum of the histogram that is output by the previous module. The module then, uses this cumulative histogram as a look-up table for calculating output values of the image.

After one frame completes passing through Image Statistics sub-module, the module outputs the histogram of the image. Equalization sub-module inputs this data. For each data that is output by Image Statistics sub-module, it sums the incoming histogram data with the cumulative sum data of the previous gray level and writes is to the cumulative sum data of the next gray level. It should also be noted that Block RAM

for keeping cumulative histogram needs to operate at the same frequency that Block RAM for keeping histogram operates at, whereas the counter for addressing the RAM block operates at the same frequency that pixel data is being captured.

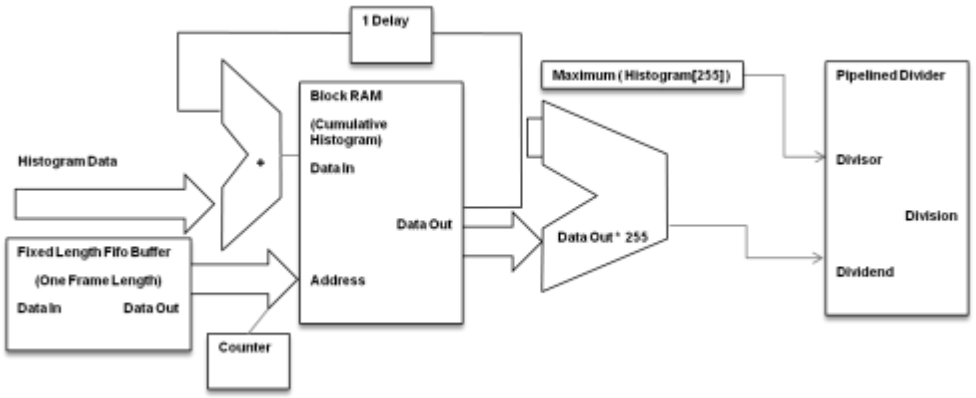


Figure 4.18 Cumulative histogram calculation and look-up circuitry.

Similar to Image Statistics sub-module, Equalization sub-module contains two RAM blocks. At the beginning of a frame, first RAM block begins to calculate cumulative histogram, while second block processes remainder of the previous image. As the calculation of the cumulative histogram completes, processing of the previous image completes too. After this point, first RAM block begins to process the image, while the other RAM block goes to sleep. The roles of these blocks reverse with the beginning of the incoming frame. Figure 4.19 summarizes the timing of these RAM blocks.

RAM 1	Calculate CDF	Process Image 1		Sleep
RAM 2	Process Image	Sleep	Calculate CDF	Process Image 2
Images	Image 1		Image 2	

Figure 4.19 Timing of two RAM blocks in histogram equalization circuitry.

Again, similar to the Contrast Stretching module, the pixel data is stored in a frame buffer after passing through the Image Statistics sub module. In this module however, the buffer needs to be longer than a frame, by the number of the gray levels. This

additional length is necessary in order to compensate for the delay that occurs during the transfer of histogram data from Image Statistics sub module to Equalization sub module.

Cooperation between the RAM blocks Image Statistics and Equalization sub-modules are summarized in the 4.20.

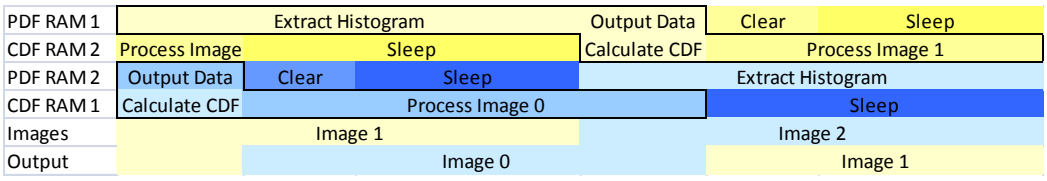


Figure 4.20 Co-operation of two RAM blocks in histogram extraction circuitry and two blocks in histogram equalization circuitry.

As the image pixels arrive to the module, the data first passes through Image Statistics sub module first RAM of the module extracts the histogram of the image. At the same time second RAM of the module transfers its contents to next module, after this transfer its contents are cleared and it's put into sleep mode. Simultaneously, the pixel data are also written into frame buffer.

At the end of the frame, Image Statistics module finishes extracting its histogram, and begins transferring it. The complementary RAM Equalization sub module takes in the histogram data, while at the same time calculating a cumulative histogram with it. When calculation of cumulative histogram is completed, pixel data of the current image is output from the frame buffer. At this point, the RAM related with cumulative histogram begins to behave as a look up table. With each pixel output from the frame buffer, the RAM outputs the corresponding cumulative histogram output, which is then transferred to pipelined divider for the final division operation. Figure 4.18 illustrates cumulative histogram calculation and look-up circuitry.

4.3.5 Plateau Method: Wang et al.

Plateau method proposed by Wang et al. was implemented as an extension to the Histogram Equalization module. The module uses the same Image Statistics and Equalization sub modules as classical histogram equalization. In addition to those sub modules, a Plateau sub-module is introduced for the implementation of this method. Placement of plateau sub-module is illustrated in Figure 4.21

The operation of this module begins with extraction of histogram of the incoming image by the Image Statistics sub module. At the end of a frame, different from the classical histogram equalization, the histogram is passed through the Plateau sub module, before reaching the Equalization sub module.

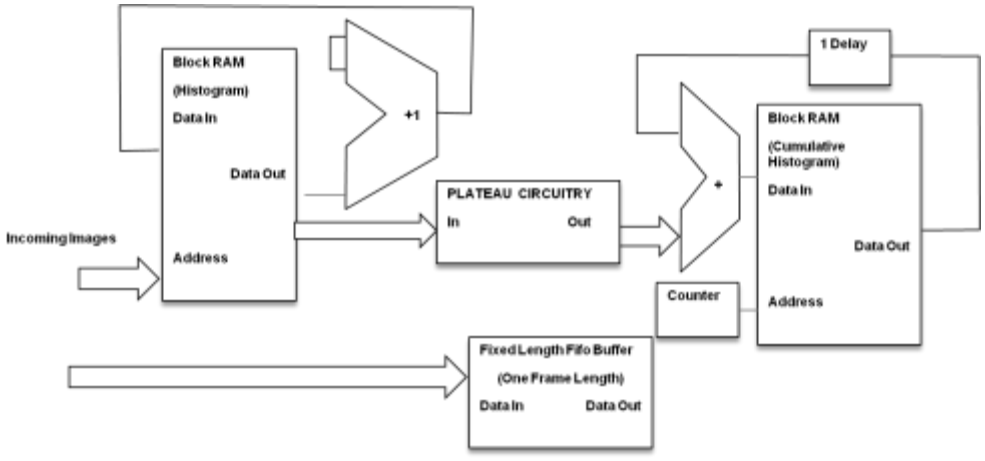


Figure 4.21 Imlementation of plateau methods.

The Plateau sub-module consists of four parts: maxima detection, histogram buffer, sorting network, and output thresholding. With the beginning of the input of the histogram into the module, maxima detection circuitry detects the peaks in the histogram. At the same time the histogram data is stored into the histogram buffer,

which is a FIFO type buffer. Sorting network, then sorts the peaks from greatest to smallest. Finally, the thresholding circuitry, thresholds the histogram by the median of the peaks. Figure 4.22 illustrates the operation of this module.

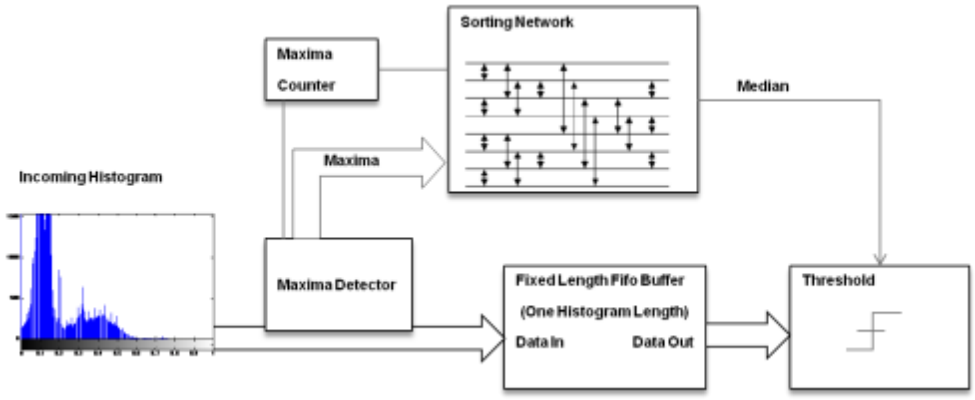


Figure 4.22 Imlementation of plateau method proposed by Wang et al.

Maxima finder circuitry detects maxima by first, calculating the first derivative of histogram, and then applying two criteria that Wang et al. [15] proposed. The derivative is calculated by simply subtracting each sequential elements of the histogram. After it was calculated for each value in the derivative it is checked whether the derivative of each element is smallest between its nearest neighbors, and whether the derivative of the previous element is negative whereas the derivative of the next element is positive.

Each time a maxima is detected it's recorded into a register array. The index of the array to be written on is determined by a counter which is also incremented. Also all the values of the array is reset to zero each time the maxima detection begins to ensure the residue of previous frames don't affect the current threshold calculation. These zeros will stay at the bottom during the sorting operation and won't affect the result.

The array that's holding the maxima is input to the sorting network. The sorting network consists of a matrix of compare and swap operations and it behaves like an equivalent of bubble sorting algorithm in programming. At each layer of the sorting network the neighboring elements of the array that hold maxima values are compared and swapped if the one in the higher index is larger than the one in lower index. After a number of iterations the items in the array will be ordered from largest to smallest.

The number of iterations required for complete sorting is equal to the number of the items of the array. For the sake of implementation modularity maximum number of items is limited to a fixed number. During the software prototyping of the algorithm maximum number of maxima that is observed was around 20 for most of the images. With a 50% safety margin the limit is set to be 30. Yet, this matrix of compare and swap circuitry consumes a considerable amount of FPGA space, and it can be reduced parametrically.

The final modification of histogram is thresholding. The threshold is found as the median of the maxima, which is the value at the output of the sorting network with the index that's the half of the value in the counter that counts the number of the maxima. At the time that threshold calculation operation is complete the histogram buffer begins to output the histogram data. For each value of the histogram if it's larger than the threshold it is set to the value of threshold.

After thresholding, the modified histogram is output from Plateau sub module and input to Equalization module. In the Equalization sub module this output is treated the same as the output of the Image Statistics sub module. Also, the length of the frame buffer is larger than classical histogram equalization method by the delay of sorting network.

4.3.6 Plateau Method: Lai et al.

Plateau method proposed by Lai et al. is implemented in a similar manner to plateau method of Wang et al. The implementation again, consists of an Image Statistics sub module, a Plateau sub module and an Equalization sub module. Similar to previous implementation, Plateau sub module inputs the histogram data from Image Statistics sub module and outputs the modified histogram into the Equalization sub module.

Plateau sub module consists of four main parts: maximum detector, histogram buffer, square root calculation circuitry, and pipelined divider. With the input of the histogram data, the maximum detector finds the maximum value of the histogram by comparing the value of each element in the histogram to its content. At the same time, the histogram data is recorded in to histogram buffer which again, is a FIFO buffer. The values of the histogram are then passed through a division operation, and then through a square root operation for an approximation of the equation (4.11). Finally, the resulting modified histogram is output to be used for equalization operation. Figure 4.22 illustrates the operation of this module.

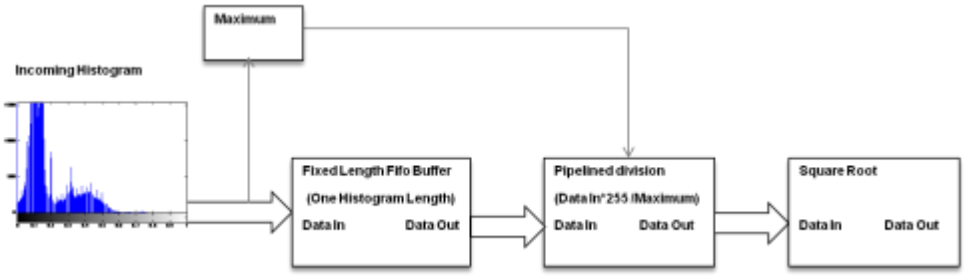


Figure 4.23 Imlementation of plateau method proposed by Lai et al.

This method also requires a frame buffer, long enough to compensate for the delay of pipelined division and square root operations.

The division operation uses a pipelined divider. Since the divider is a fixed point implementation, any multiplication operations need to be implemented before the division. For this method specifically, since there is a square root after division, the value to be divided is multiplied with the square of the number of gray levels.

The challenge with implementing this method is to implement the power operation in (4.11). A very close approximation for it might be cube root approximation. There are numeric methods for Newton – Raphson (4.12) iterations or Hailey (4.13) iterations such as:

$$n(i) = \frac{1}{3}\left(\frac{x}{n(i-1)^2} + 2n(i-1)\right) \qquad (4.12)$$

$$n(i) = \frac{1}{3}\left(\frac{n(i-1)^3+2x}{2n(i-1)^3+x}\right) \qquad (4.13)$$

However, these operations require additional dividers, which consume too much FPGA space to be implemented. Using square root approximation wouldn't give results as good as a cube root approximation, but square roots can be calculated algorithmically via a series of shift and subtract operations, and it will consume much less space. Therefore, square root approximation was selected for implementation.

Square root circuitry was implemented in a pipelined manner. It inputs the result of the division circuitry and outputs the result of the equation (4.11). The output is sent to Equalization sub module where according to this output, cumulative histogram calculation and look-up operations are performed.

4.3.7 Contrast Enhancement Module

All of these implementations mentioned above have different advantages and disadvantages depending on the situation. In some situations one may give a more useful output than the others, while in some other situations using another may be more advantageous. For example histogram equalization would improve the visibility of the region of interest, when the number of pixels of this region is larger than other regions. On the other hand, when it is smaller than the other regions a plateau method would work to balance the disproportionately low gains for the region of interest. Other times, there may be situations that may require preservation of the proportionality between the values, these situations would require contrast stretching. So, it would be more useful to implement the mentioned methods in a module in which the method can be selected by the user, during the run-time rather than implementing a single module.

Implementing a contrast enhancement module that wouldn't use too much additional resources since the modules have a lot of similar properties. First of all, all the implementations require a frame buffer, though with different lengths. They also require at least one division operation. These two modules use the most space in all of

the implementations. In addition to that, histogram based methods share the Image Statistics and Equalization methods. Those overlaps allow a compact design that includes all the methods in a single module.

4.3.7.1 Structure of the module

The module has four modes that are controlled via an outer select signal. Depending on the mode selection, enable signals of the modules are set or reset, and inputs and outputs are multiplexed to necessary sub-modules.

The module is structured to share the frame buffer in all modes, so it's enabled regardless of the select signal if main module is enabled. It allows either using an external memory or internally allocating a Block RAM which is selecting in the synthesis-time. Different methods require different FIFO buffer sizes. To handle this, the controller dynamically selects the length of the buffer according to the select signal. Figure 4.24 illustrates the structure of the module.

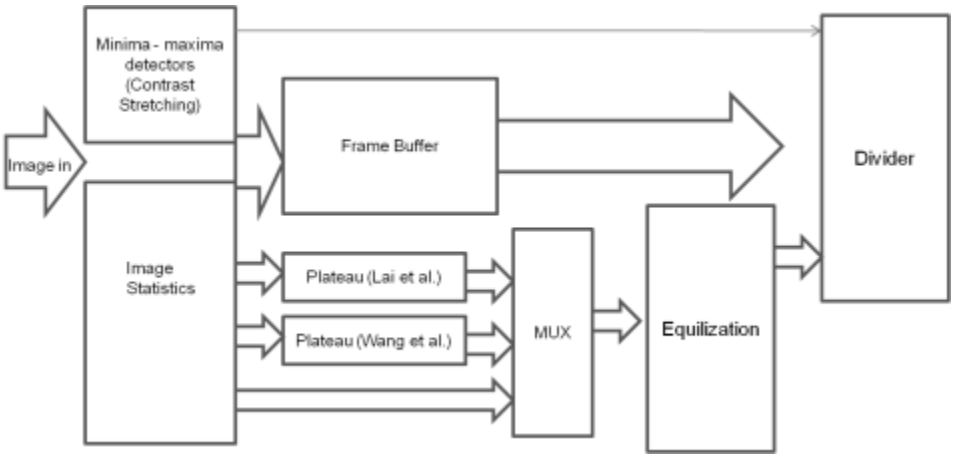


Figure 4.24 Architecture of the contrast enhancement module.

The pipelined divider consumes the most area by comparison to other circuitry so it's also shared between the methods. At any given time, maximum number of division operations is two, therefore two pipelined dividers are implemented. One of the pipelined dividers is shared by all the methods, whereas the other is specifically allocated for the Plateau sub module of the method proposed by Lai et al.

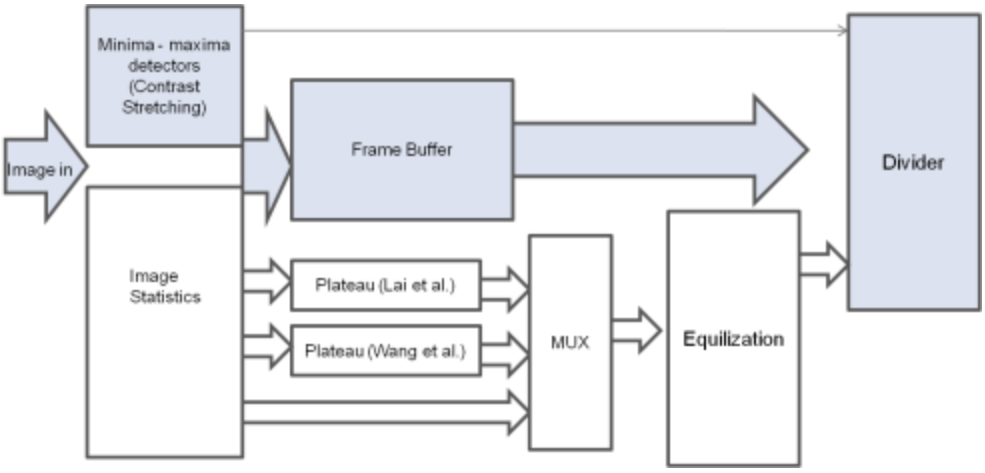


Figure 4.25 Contrast enhancement module in *Contrast Streching* mode.

As expected, histogram based methods also share the Image Statistics and Equalization module. Plateau methods proposed by Lai et al. and Wang et al. are implemented between these two modules. In addition, for the Contrast Stretching method an additional maxima and a minima finder is implemented.

When the select signal indicates that current mode is Contrast Stretching, minima and maxima finder is enabled, frame buffers length is set to an exact frame length and inputs of the pipelined divider is connected to the output of the maxima and the subtraction of the minima from the frame buffer. Figure 4.25 illustrates the active parts in contrast stretching mode.

When, on the other hand, select signal indicates histogram based methods Image Statistics and Equalization circuitry is enabled, frame buffers length is set to the addition of the delay of the selected method to one frame, and inputs of pipelined divider is connected the look up result from and the maximum value of the cumulative histogram. Figure 4.26 illustrates the active parts in histogram equalization mode, and Figure 4.27 illustrates the active parts in histogram equalization with plateau modes.

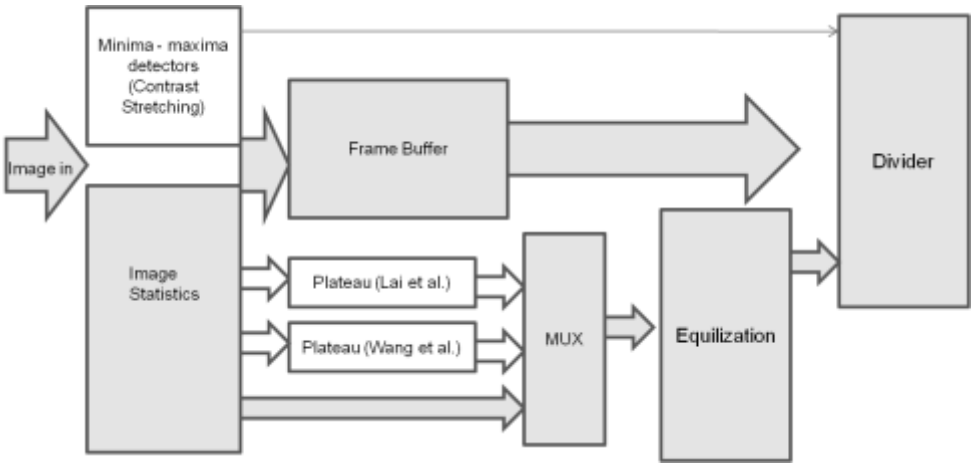


Figure 4.26 Contrast enhancement module in *Histogram Equalization* mode.

The control logic at the Contrast Enhancement module disables the unused modules, this way it doesn't consume any extra power. All the modules are parametrically connected as sub modules the main module. This allows the main module to keep genericity of the sub modules at a top level.

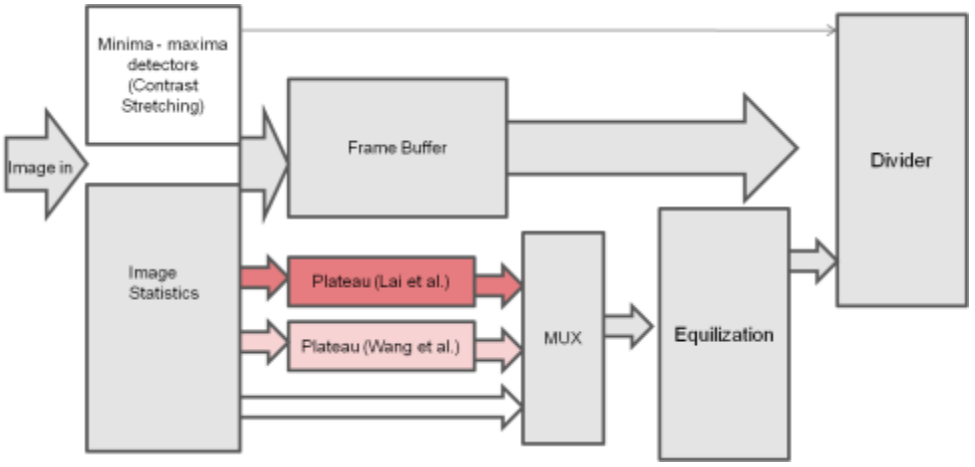


Figure 4.27 Contrast enhancement module in *Histogram Equalization with Plateau* mode .

CHAPTER 5

NOISE REDUCTION

This chapter discusses sources of noise in the image, and algorithms to reduce the noise. First, Section 5.1 states the sources of noise, which gives hints about the characteristics of the noise, and then Section 5.2 discusses three noise reduction methods, namely spatial filtering, wavelet based methods, and time domain filtering. Finally, Section 5.3 provides the implementation of the spatial filtering method.

5.1 Sources of the Noise

The image is produced from the incoming infrared radiation, as a result of a complex process. First, the radiation falls onto an active material, and then the material responds to incoming radiation with changing resistance or diode voltage. After that this change is read out by an electronic circuitry. Finally, the output of the readout electronics is converted to a digital value via an analog to digital converter. All of these steps introduce some level of noise to the image.

The noise on the image may be a source disturbance in the user experience. Disturbance may vary from a simple discomfort for the user to leading user to miss or discard important details. In order to get around this, well known methods for noise reduction can be employed.

It was observed tjat the noise on the image was evenly distributed in time domain. On the other hand, it may have some space domain characteristics as a result of readout methods. The algorithm that readout circuitry uses for multiplexing and outputting pixel data, can lead relatively low frequency noise to appear as patterns on the image.

In METU MEMS Research and Applications Center, the readout circuitry of the infrared detectors is designed in a way that reads out the pixel data in the horizontal direction. For that reason the noise on the image appears as horizontal lines. This knowledge about the spatial characteristics of the noise makes it easier to develop and use spatial filters, specifically designed for this case.

5.2 Noise Reduction Methods

5.2.1 Spatial Filtering

Since the noise on the image appears mostly as horizontal lines, a low pass filter with vertical components can be employed to suppress it. This filter will smooth line to line changes in the image, reducing the effect of the noise on the image. The filter can be applied using a convolution operation. This is a simple and fast solution and can be implemented within relatively small FPGA space. It provides a smooth image and reduces the noise on it. However, this method has some drawbacks. It does not suppress the noise as much as other methods. In addition to that, it loses too much detail in the vertical direction. This may be unaffordable in a small image. Figure 5.1 provides an example of spatial filtering.

Figure 5.1 a) Raw image, b)Spatially filtered image.

5.2.2 Wavelet Based Methods

Another approach for reducing the noise is using wavelets. As the noise appears as horizontal lines, there is vertical spatial frequency that is associated with it. Using this property, noise can be suppressed more effectively without filtering out too much detail.

There are examples in the literature that use wavelet based methods for noise reduction [22-24]. Generally, images are analyzed into their low pass and high pass images, after that, high pass images, which contain the noise component are filtered. A simple and effective way to free these high pass images from noise is to threshold them. With thresholding weak details, which are mostly noise, much of the noise is eliminated. Figure 5.2 and Figure 5.3 illustrate wavelet analysis, thresholding, and synthesis.

Figure 5.2 Wavelet analysis.

The wavelet thresholding operation can be iterated further by analyzing low pass image into its low pass and high pass details. This way, noise components with lower spatial frequency can also be filtered out.

However, this method has its drawbacks too. It is calculation intense, and it requires a lot of memory. Several levels of analysis and synthesis pairs would require too much memory to be implemented on a small FPGA. In addition, the wavelet analysis and synthesis filters have different frequency responses. After aggressive filtering, frequency responses of the filters might result in patterns that are visible on the image.

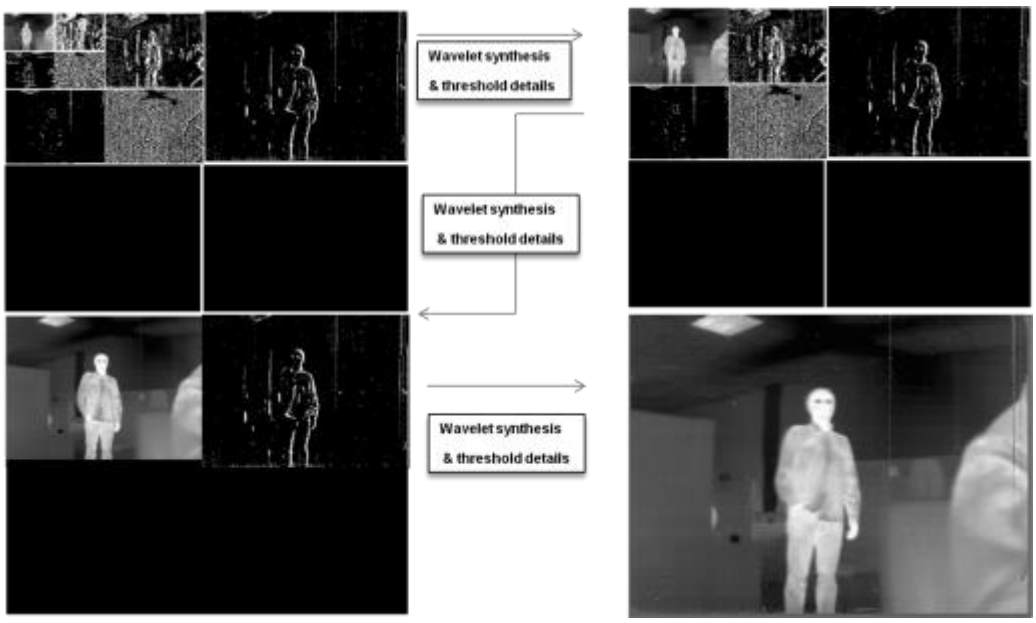


Figure 5.3 Thresholding of details and wavelet synthesis.

5.2.3 Time Domain Filtering

As an alternative to, or complimentary method for spatial filtering is time domain filtering. Sequential values of each pixel can be passed through a one dimensional low pass filter, in order to create a noise suppressed image. The kernel can be implemented in a similar manner to a space domain filter kernel and it would be simple and fast. Figure 5.4 illustrates time domain filtering.

This method provides relatively better noise reduction performance, since it would have a lower cut off frequency than the low frequency noise that is appearing in the image as lines.

Figure 5.4 Time domain filtering.

However, this method also requires too much memory. It needs to keep several frames in the memory to be effective enough. Also, it increases the response time of the system and causes a motion blur. This may cause the system to lose some important data when fast response is necessary.

5.3 Implementation

Since there are limited resources, in terms of FPGA space only spatial filtering method was implemented. The implementation is similar to the one discussed in Chapter 3. It consists of a kernel and several FIFO buffers to hold data associated with the lines. The implementation is illustrated in Figure 5.5.

The width of the kernel is set parametrically to be a single pixel. The remaining pixels on the line are stored in fixed length FIFO buffers. The lengths of the buffers are one pixel shorter than the width of the image. The input of each buffer is connected to the output of a line, and its output is connected to the input of the next line in the kernel. The number of buffers is one less than the number of lines in the kernel.

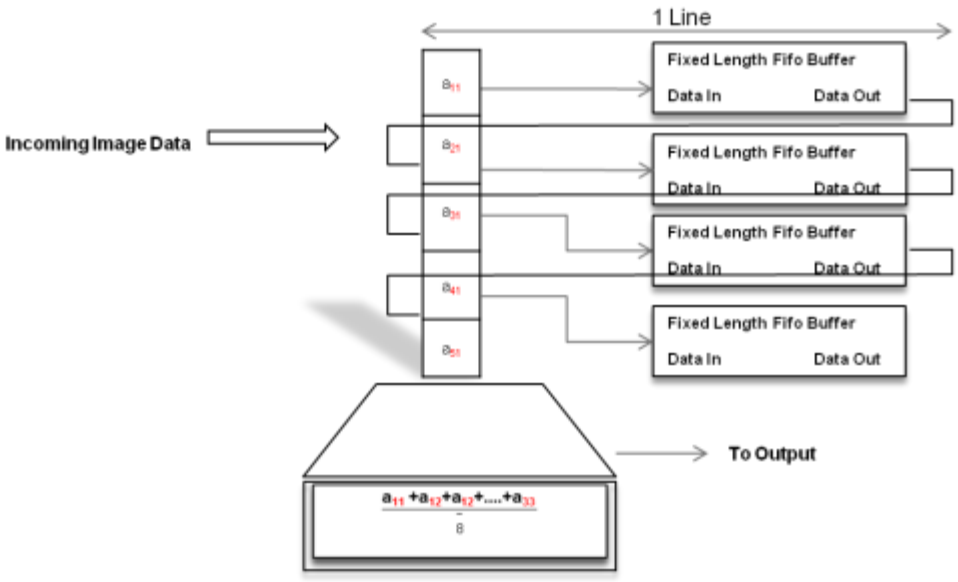


Figure 5.5 Spatial filter implementation.

In order not to add an additional divider, sum of the coefficients in the kernel are chosen to be a power of two. The coefficients are selected by MATLAB trials as {1,2,2,2,1} which has a sum of 8. The output is formed by a division operation of shifting the weighted sum of the kernel by 3 bits.

CHAPTER 6

PSEUDO - COLORING

This chapter focuses on pseudo-coloring of IR images. It discusses pseudo-coloring briefly, and then it provides some pseudo-color palettes, finally it discussions a static and a generic implementation. Section 6.1 provides an overview pseudo-coloring, and Section 6.2 presents commonly used palettes for IR images. Finally, Section 6.3 discusses implementations of these palettes.

6.1 Pseudo - Coloring

Infrared sensors respond to radiation intensity with variations in their output voltages. This voltage levels are then converted to digital values in A/D converters, and these values are put together in a display according to their relative position in order to form images. Often, the values are converted to gray levels directly, after some processing. This is a straightforward process where higher intensity signals are presented with higher intensity light in the display.

Another approach to display values with varying signal levels is to assign colors to intensity values. It is called pseudo-color or false-color since the signals represented with different colors, aren't signals with different frequencies, but rather signals with different intensity levels This technique is also employed in some areas like medical magnetic resonance imagery (MRI), satellite and radar image reconstruction, and x-ray imaging. In these fields intensity values from the sensing devices are matched to some colors, rather than intensity levels. This allows viewers a more natural experience, and

more importantly it provides the viewers with the environment in which they can use their intuitive senses.

In the nature, it is safe to assume that some objects or places with relatively high or low temperatures have linkage with certain colors, so humans evolved visual interpretation systems that make temperature meanings out of colors. By the nature of the infrared radiation, the hotter the object that is radiating infrared light, the more intense the radiation is. For this reason, matching increasing infrared radiation intensity with colors with increasing heat would make it easier for the viewer to make sense.

There is also another advantage to use pseudo color methods, which is a more efficient use of the dynamic range of the display. In a classical RGB type display, there are only as many gray levels as the resolution of a single color. However, when a pseudo color method is employed this number is multiplied by the number of color gradients used. In a sense, signal is amplified by the number of color gradients that are used.

The book in [25] discusses two main ways to apply pseudo color onto images, namely intensity slicing and intensity to color conversion. Intensity slicing is basically separating image into blocks and coloring blocks, whereas intensity to color conversion is assigning a color to every value in the image.

Intensity slicing is the method, in which the intensity of the incoming image is divided into *slices*, and then these *slices* are assigned different colors. This is particularly useful for detecting regions with large changes with respect to the environment. Intensity slicing is employed in many applications like medical x-rays and geographical imagery.

The other alternative is the generalized form of intensity slicing, which is called intensity to color conversion. In this method, RGB color values of the output are produced from the intensity of the image via three transforms. These transforms are often referred to as pseudo color palettes or scales. Palettes are generated as piecewise linear or sinusoidal functions, and colors are generated by combinations of these functions.

6.2 Pseudo-Color Palettes

One of the commonly used palettes is rainbow color scale which is named after the color distribution of a rainbow. This scale starts with blue color, then progresses to red finally, reaching yellow at the end. Figure 6.1 demonstrates this palette and an image pseudo-colored using this palette.

The same palette can be generated with a sinusoidal interpolation. This palette would give a smoother look to the image. However, it should be noted that some colors are used twice, and this might give the user a wrong impression about the temperature. Figure 6.2 demonstrates this palette and an image pseudo-colored using this palette.

Another useful palette, specifically for infrared pseudo coloring, is hot-metal scale. In this scale, colors are organized in the order that would imitate the light radiation of metals with increasing temperature. This scale starts with blue color, then progresses to yellow, and approaches to red at the end. Figure 6.4 demonstrates this palette and an image pseudo-colored using this palette.

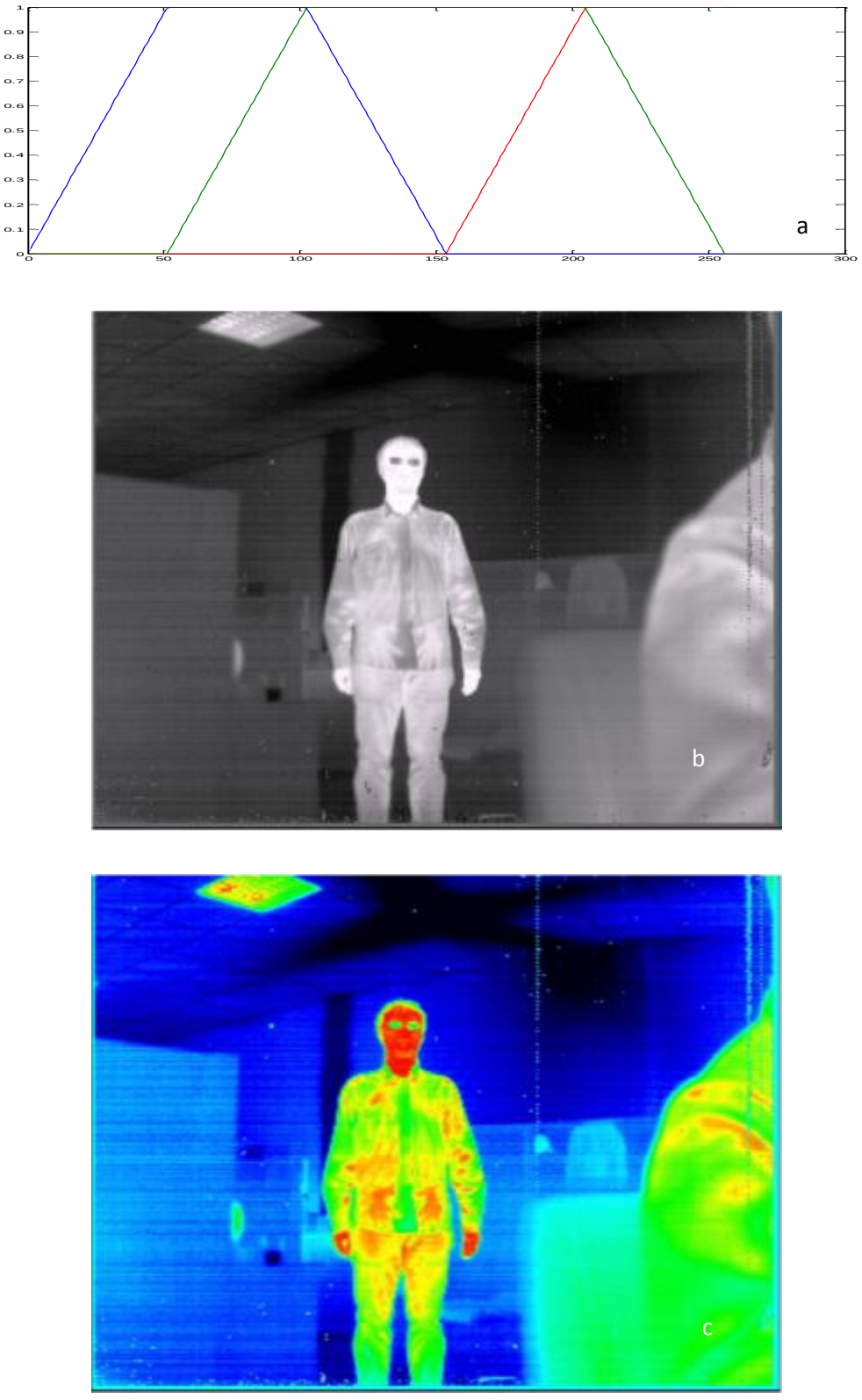
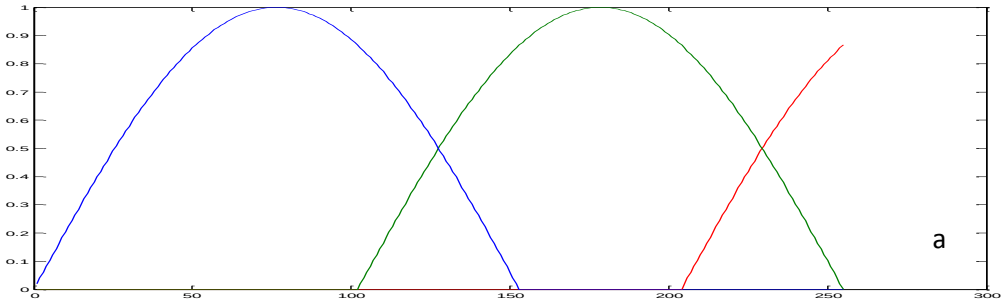


Figure 6.1 Rainbow scale pseudo coloring with a linearly interpolated palette a)The palette used for transformation, b)Raw image, c)Pseudo colored image.



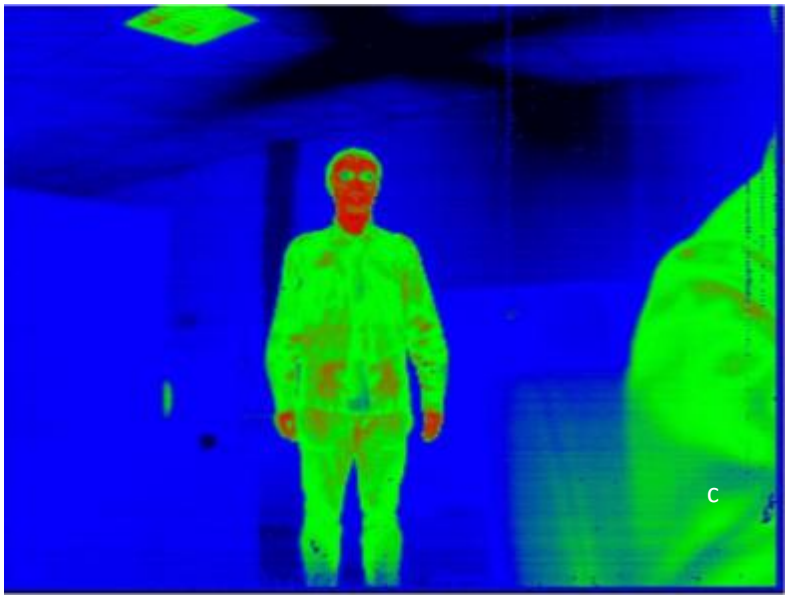


Figure 6.2 Rainbow scale pseudo coloring with a sinusoidally interpolated palette a)The palette used for transformation, b)Raw image, c)Pseudo colored image.

A last alternative that can be implemented is to directly copy the intensity level of the incoming image to a single channel, for example to green. Although it wouldn't introduce any additional visibility for details, it would provide a smoother view. Figure 6.3 provides an example of the result of directly mapping gray levels to green color channel.

Figure 6.3 a) Raw image, b) Pseudo colored image by directly mapping gray levels to green channel.

Figure 6.4 Hot-metal scale pseudo coloring with a linearly interpolated palette: a) The palette used for transformation, b) Raw image, and c) Pseudo colored image.

Figure 6.5, Figure 6.6, and Figure 6.7 provide comparisons of the pseudo-color palettes discussed in this section.

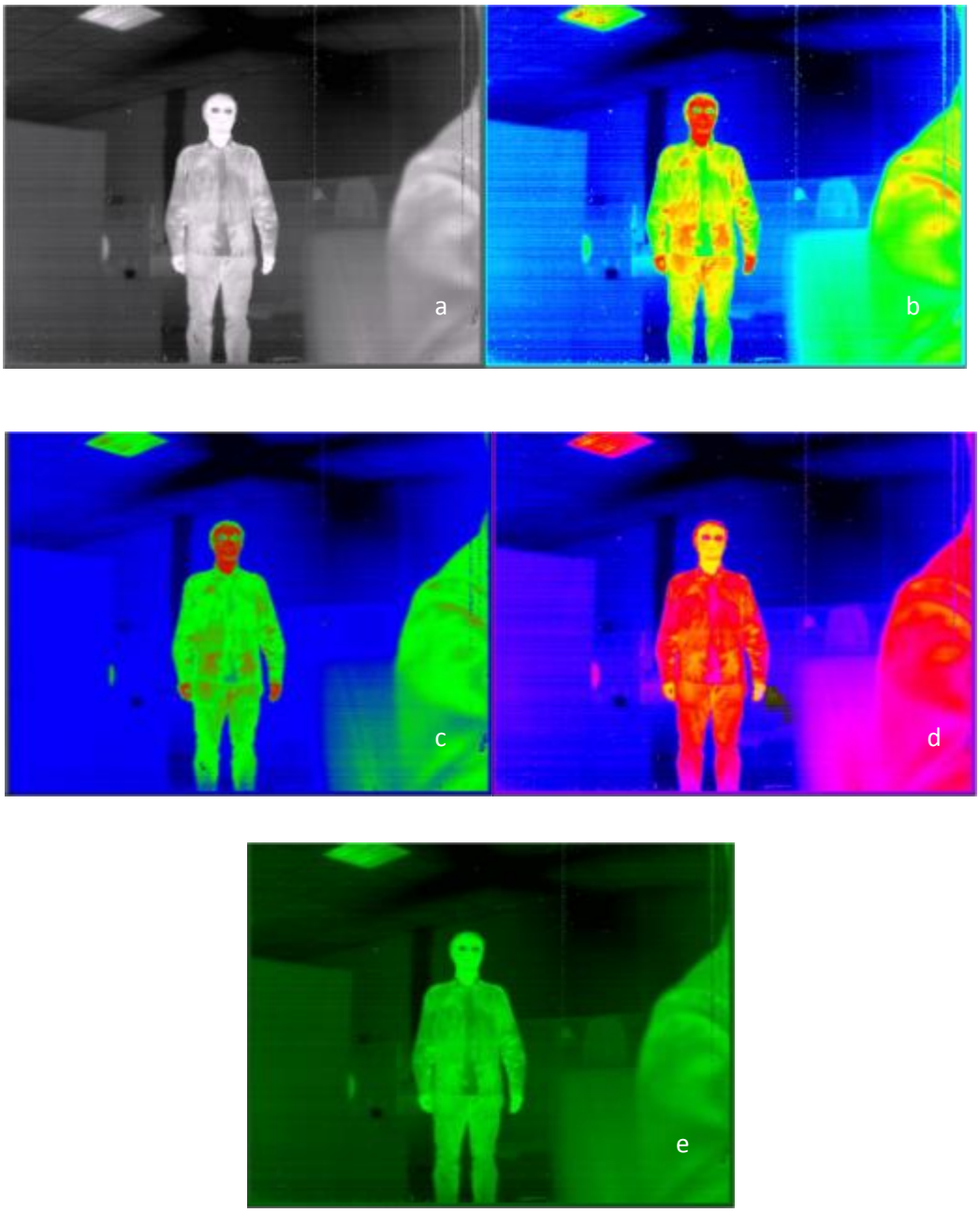


Figure 6.5 Results of pseudocoloring methods: a) Raw image, b) Pseudo colored image rainbow scale (linear), c) Rainbow scale (sinusoidal), d) Hot metal scale, e) Direct green mapping.

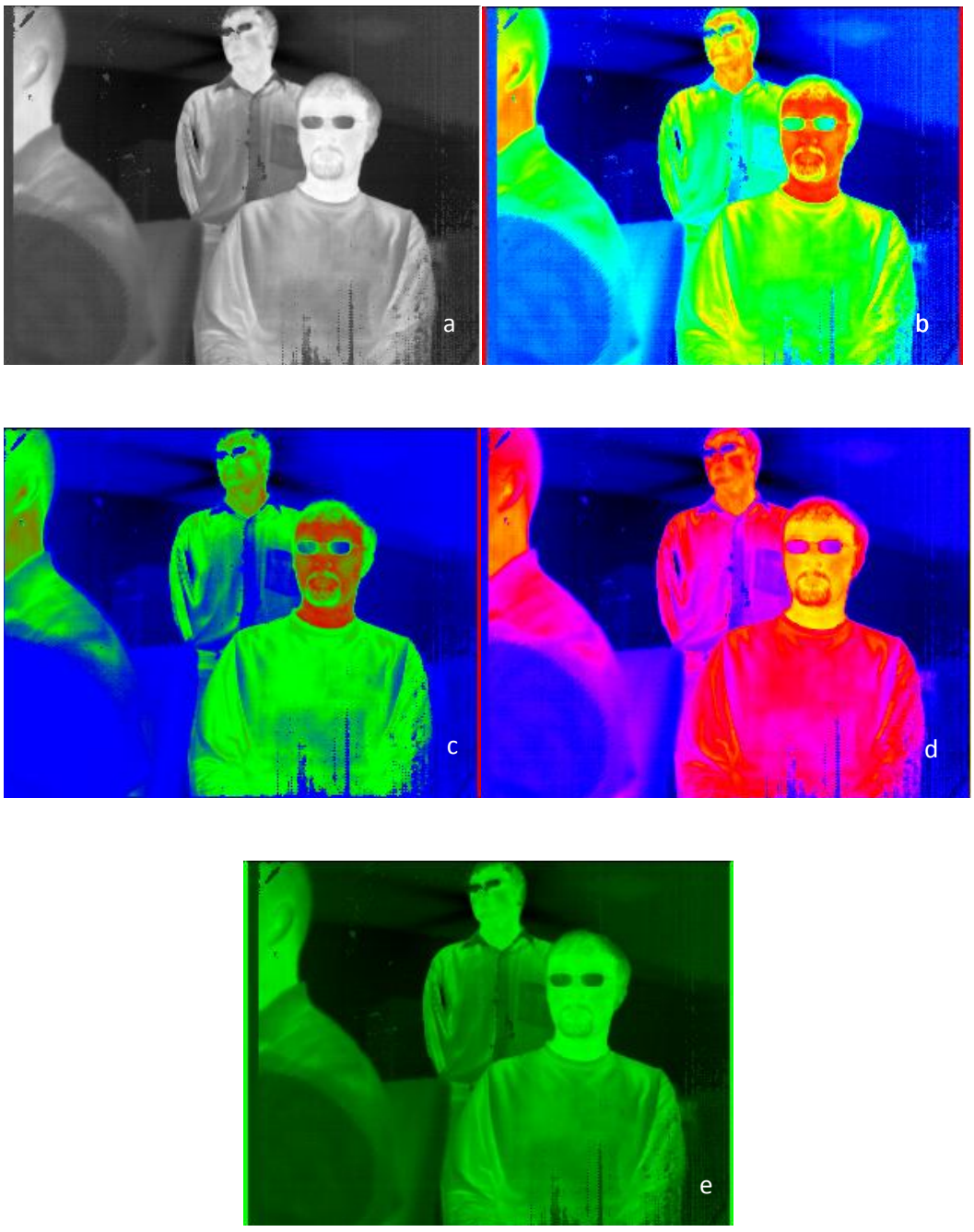


Figure 6.6 Results of pseudocoloring methods: a) Raw image, b) Pseudo colored image rainbow scale (linear), c) Rainbow scale (sinusoidal), d) Hot metal scale, and e) Direct green mapping.

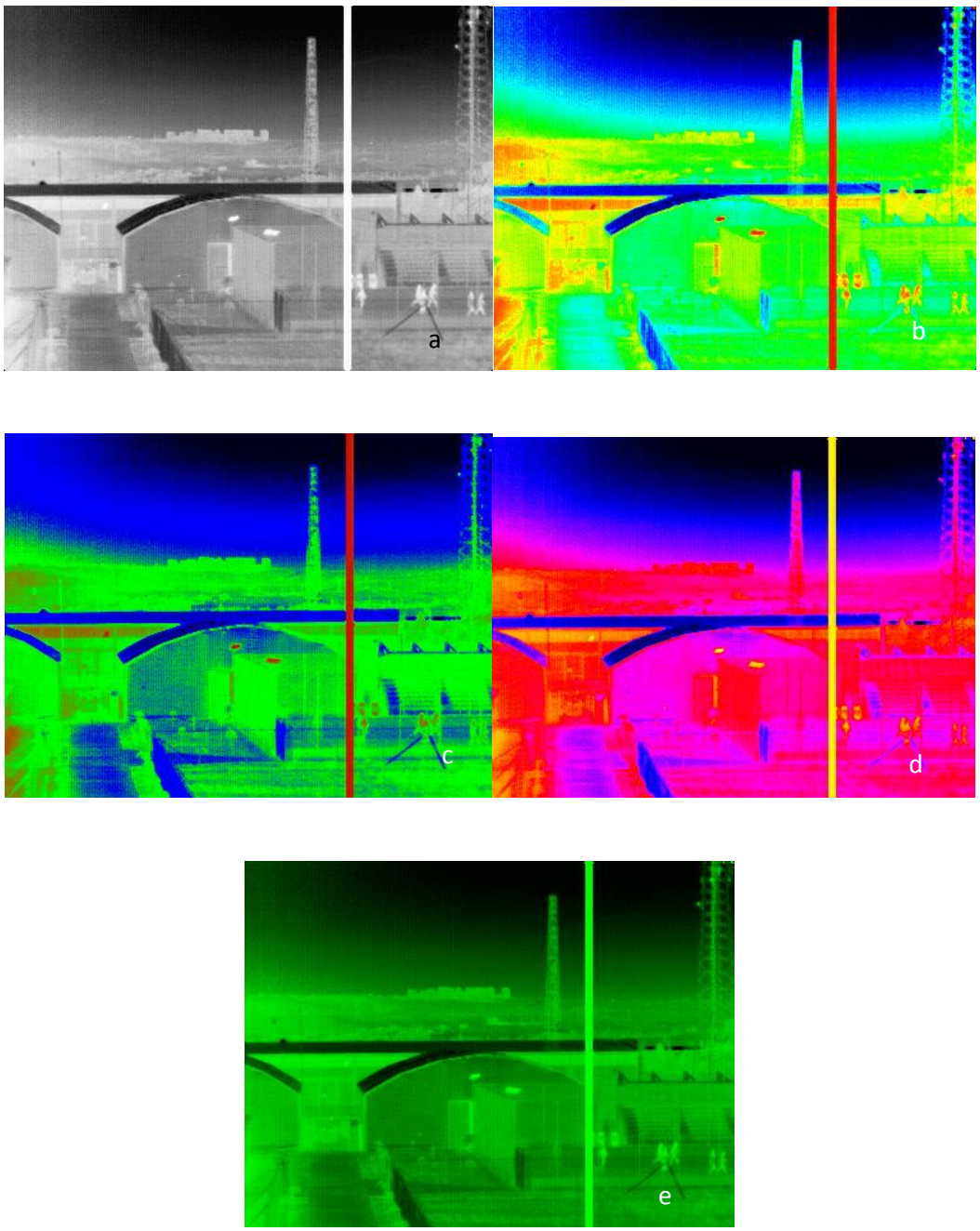


Figure 6.7 Results of pseudocoloring methods: a) Raw image, b) Pseudo colored image rainbow scale (linear), c) Rainbow scale (sinusoidal), d) Hot metal scale, e) Direct green mapping.

6.3 Implementation

Pseudo color methods are implemented for two possible uses. The first implementation assumes a fixed color resolution, which is true for almost all displays. It builds on this assumption and uses a fixed implementation, which is both small and fast. The second implementation, on the other hand employs a generalized implementation. It is flexible, generic, and it can be adapted to various color resolutions. However, it uses more FPGA resources and it's slower.

Most color displays are designed with 8 – bit color resolution per color channel. This knowledge is useful when building fixed coloring implementations. For pseudo color transformations, when the size is fixed, look-up tables can be used instead of calculation circuitry since the values of the transformation functions wouldn't change as well as the resolution.

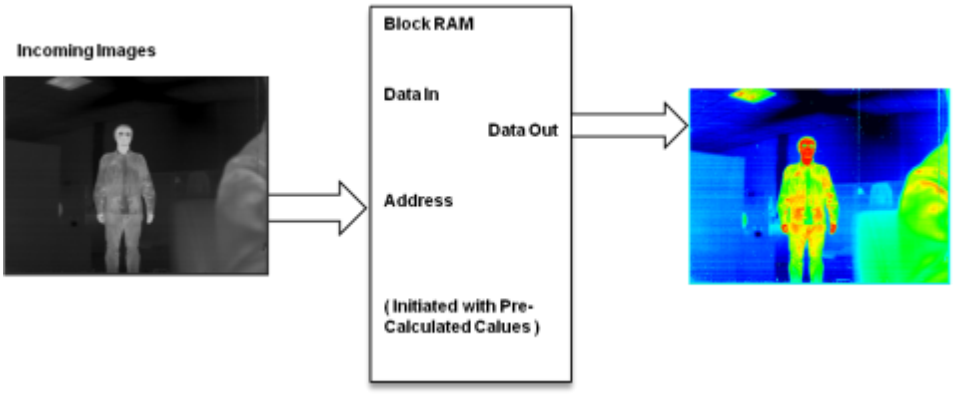


Figure 6.8 Pseudocolor implementation by using Block RAM as look up table.

This method implements a look-up table using Block RAMs in FPGA. Xilinx FPGAs allow presetting of Block RAM data during programming. It also allows a "Read Only" mode, in which contents of the RAM cannot be changed during run time.

Implementation includes three RAM blocks corresponding to red, green, and blue channels. RAMs are divided into sectors for look up values of different palettes. The lookup values in these RAMs are calculated and initialization files were generated with an outside software. Address inputs of the RAMs are used as input. With the selection of the palette an additional offset is added to the input that addresses the corresponding sector for the palette. Outputs of the RAMs are used as the final output of the module. Figure 6.8 illustrates this Block RAM based implementation.

Second implementation is more generic. The idea behind this implementation is to implement the point wise linear transformation equations for three color channels. It inputs the color resolution and the number of intensity levels as parameters. Then it divides the number of intensity levels to the number of colors and generates regions in synthesis time. Figure 6.9 illustrates this generic implementation.

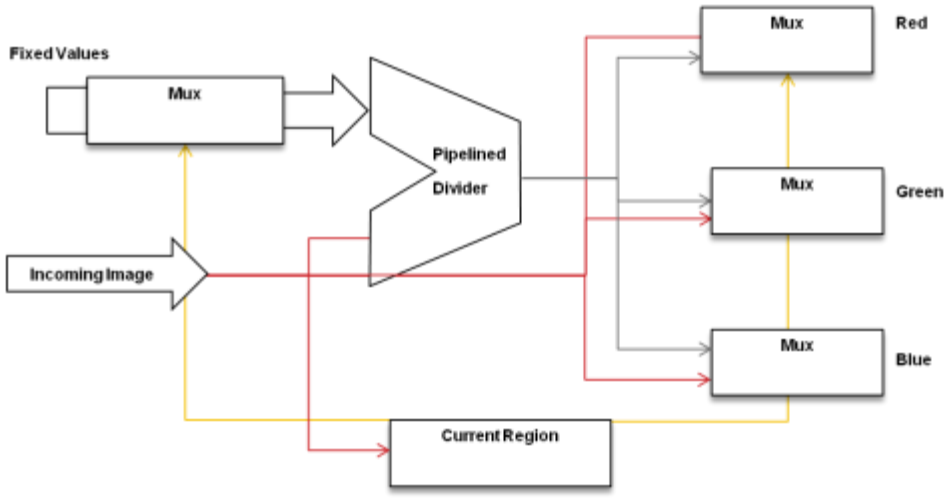


Figure 6.9 Generic implementation of pseudo coloring algorithm.

During the run time the module decides which region the input image value falls into and calculates output in accordance with the point wise linear transformation function. At each region only one division is necessary, so only one divider would be enough for all the calculations. Finally, the multiplexers at the output would decide which value to be output for each color channel according to the region that incoming image data falls into.

CHAPTER 7

RESULTS

This chapter provides the verification of the camera electronics, and results of the implementation of the image processing algorithms. The testing and verification of camera electronics were done by simple measurements. The testing of the implementation of the image processing algorithms, on the other hand were done by sending data to, and receiving data from the FPGA. The camera electronics are fully functional, and image enhancement software fits the frequency and the area constraints. Section 7.1 discusses verification of the camera electronics. Section 7.2 presents the results of image enhancement.

7.1 Camera Electronics

The camera the hardware requires the testing of the biasing and the programming of board components and the testing of data acquisition. Correct biasing and programming of the hardware was tested and verified via simple measurements. Furthermore, correctness of data acquisition is verified by the comparison of the output of the scope mode of the software with oscilloscope measurements.

D/A converters on the camera board were tested by sending values of 0V (minimum), 5V (maximum), and random values in between, measuring the output with a multimeter. The SPI programming module that programs the IR detector array was verified by observing its output with an oscilloscope.

The available A/D converter was tested by giving random values to its input via an external voltage supply and observing the scope window of the camera software. Finally, tagging module of the data acquisition was tested and verified by comparing corner pixels of the image with the corner pixels of the detector array.

7.2 Image Enhancement

7.2.1 Test Setting

For testing, the XEM 3010 evaluation board from Opal Kelly was used. The board includes a Spartan 3 series XC3S1500 FPGA, which was chosen to be the platform for the designs. It includes an additional micro controller unit (MCU) for USB communication, a PLL for clock generation, and an SDRAM for data storage. Figure 7.1 demonstrates Opal Kelly XEM 3010 board.

The board connects to a host computer through USB port, where interfacing software handles data transfers to and from the board. Data transfers to and from the FPGA are handled as *endpoints*, in the standard libraries provided by Opal Kelly. In terms of behavior, Opal Kelly libraries provide three types of endpoints. *Trigger* type endpoints when activated from the complimentary side stay high for one clock duration. *Wire* type endpoints, on the other hand keep the data that it is programmed with until the complimentary side changes it. The third type is for bulk transfers of data, and it is called *pipe*.

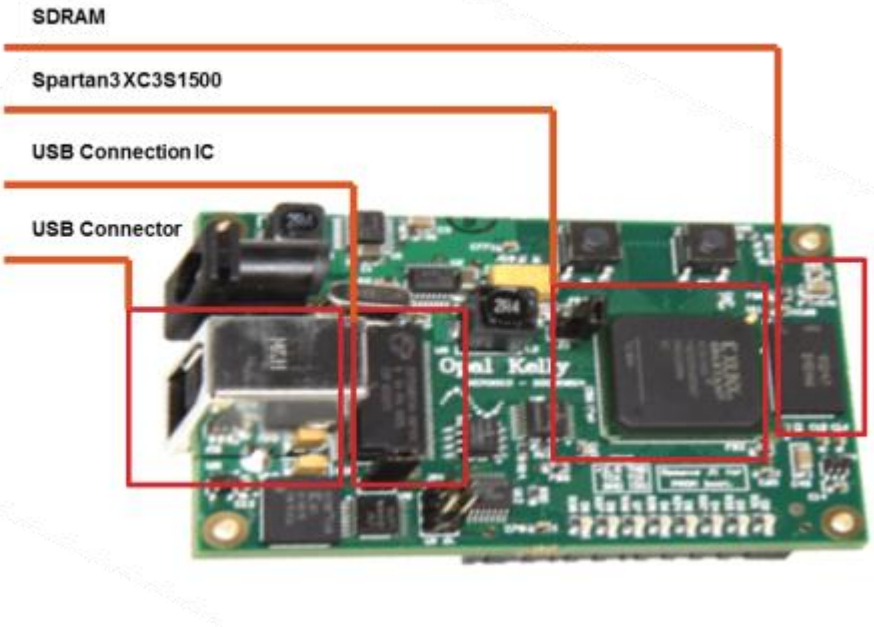


Figure 7.1 Opal Kelly XEM3010 Board.

For this test, trigger type endpoints were employed for synchronization signals such as the beginning of a new frame or change of a signal. Wire type endpoints were employed for selection signals such as threshold of defective pixel correction or type contrast enhancement method. Finallyimage was transferred through pipe type endpoints into and out from the FPGA to the software.

Although it is possible to combine them, separate methods such as defective pixel correction or contrast enhancement were tested separately. First, the host computer sent the image data to the FPGA. After that the FPGA processed the image with the relevant method. Finally the host computer transferred back the result back from the FPGA in order to display or save it.

Throughout the testing clocking frequency was kept constant at 2 MHz, which was the desired frequency for the systems to work with. It is hard to determine whether the errors are occurring due to clocking circuitry with higher frequencies or not. They can settle with or, due to round off or sign errors in the arithmetics by looking at the results only. For this reasons performance characteristics of the circuits were determined by simulation results.

The Xilinx ISE design suite provides accurate post-route simulations for characterization of clocking performance. It provides a 2ns resolution for clock signal generation, 1 ns for on and 1 ns for off time assuming symmetric clocking.

For verification of the proposed implementation, the RMS difference between the results of the hardware and software implementations and the MATLAB prototype was compared. The measure was taken to be the second order norm of the difference of two images. For cancelling out effects of shifting the image to be compared was shifted as many times as necessary and minimum of the resulting differences was taken. The result was taken as parts per thousand ($1/10^{-3}$)

7.2.2 Results

7.2.2.1 Defective Pixel Correction

The first method to be tested was the defective pixel correction algorithm. FPGA implementation of this algorithm produced almost identical results as the MATLAB prototype. This was expected since there are no extensive arithmetical calculations in the algorithm. In the median filter there are only complete pixel value replacements so it wouldn't produce results with round of errors. Results of the test verified this expectation the deviance of the results from MATLAB prototype was 0.0×10^{-3}.

The clocking performance of the circuitry was quite above the target frequency. It performed without any problems up to 20 MHz, ten times the desired frequency. After this frequency shift registers in the kernel began to produce timing violations. Figure 7.2 demonstrates the result of defective pixel correction and its comparison to MATLAB prototype.

Figure 7.2 Defective pixel correction results: a) Original image, b) MATLAB prototype, c) Output of the device.

7.2.2.2 Contrast Enhancement

For the contrast enhancement results are satisfyingly similar to the results of MATLAB prototypes except for plateau method proposed by Lai et. al. In other methods the error can be associated with the round off errors in the fixed point operations. The RMS errors of contrast stretching and histogram equalization were 0.2×10^{-3} and 0.6×10^{-3} respectively. Figure 7.3 demonstrates the result of contrast stretching, and Figure 7.4 demonstrates the result of histogram equalization.

Plateau method proposed by Lai et al. is performing relatively poorer than the MATLAB prototype. This quality drop is due to the square root approximation used in place of the power of 0.31 operation proposed by the article. Its deviance from MATLAB prototype was 2155.4×10^{-3}. Figure 7.6 demonstrates the result of this method and its comparison to MATLAB prototype.

Plateau method proposed by Wang et al. suffered discontinuities and caused flashes when the maximum number of the maxima that can be kept in the memory fell below 25. However over this number the method functioned properly. Its deviance from MATLAB prototype was 23.6×10^{-3}. Figure 7.5 demonstrates the result of this method and its comparison to MATLAB prototype.

Figure 7.3 Contrast stretching method results: a)Original image, b) MATLAB prototype, c) Output of the device.

Figure 7.4 Histogram equilization results: a) Original image, b) MATLAB prototype, c) Output of the device.

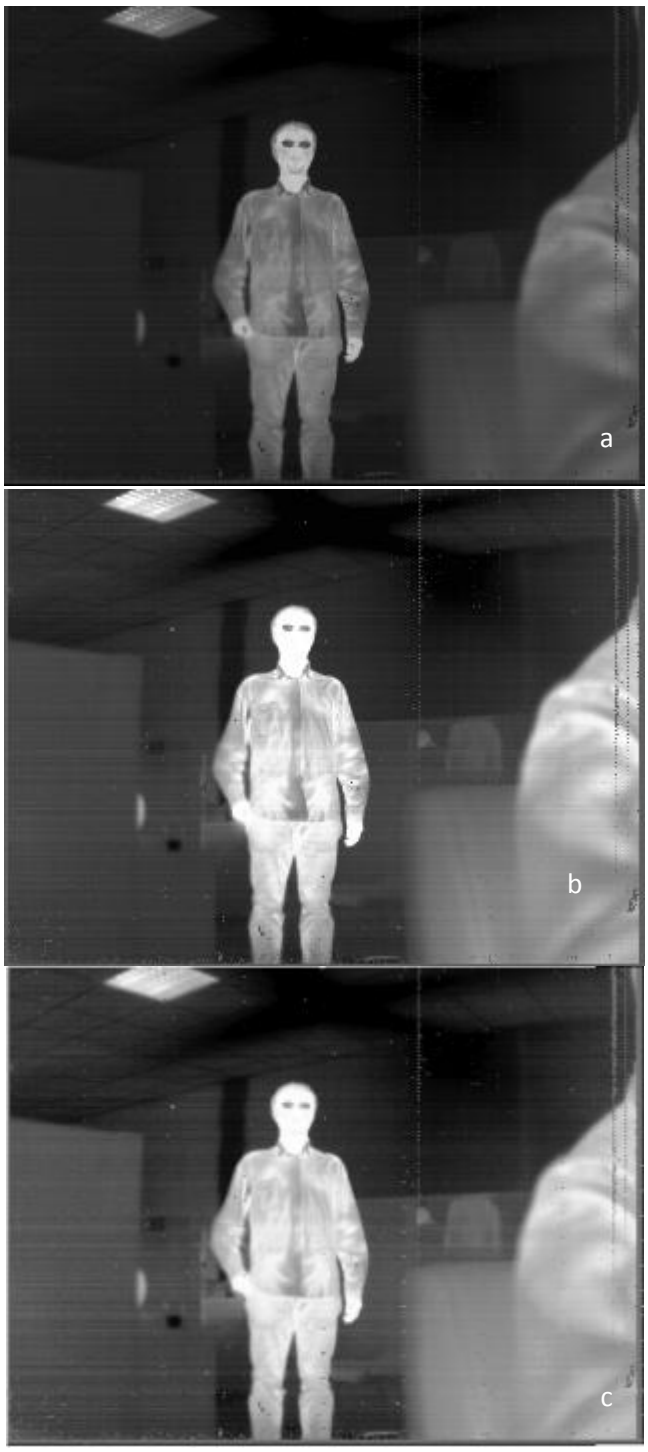


Figure 7.5 Results of plateau method proposed by Wang et al. a) Original image, b) MATLAB prototype, c) Output of the device.

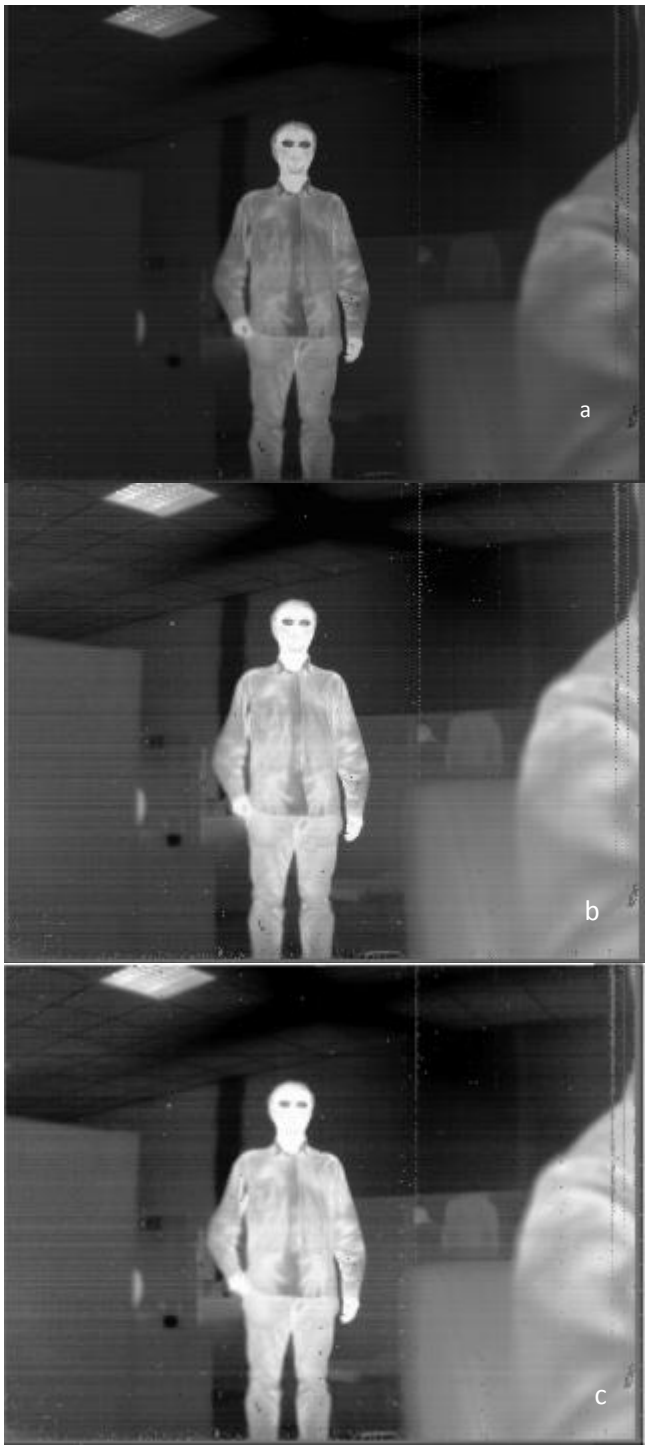


Figure 7.6 Results of plateau method proposed by Lai et al.: a) Original image, b) MATLAB prototype, c) Output of the device.

The major limiting factor frequency in these methods is the pipelined divider. Since all the methods require at least one division operation, their operating frequencies are limited with the bottleneck of the system which is division. The maximum frequency that could be achieved with these methods was 3.3 MHz. It's faster than the desired frequency with a safe 65% margin.

The cause of the bottleneck in the division circuitry is the asynchronous path in subtract and compare operation. This path can be retimed to have a better clocking performance. However such an operation will increase the FPGA space requirements.

Of all the image processing operations discussed here, only contrast enhancement module came close to exceed the area limitations. Whereas the other operations used a few RAM blocks and a small percentage of the logic, contrast enhancement module consumes 25% of RAM blocks 93% of logic slices in 12 – bit operation, and it uses 100% of RAM blocks and 98% of logic slices in 14 bit operations.

7.2.2.3 Noise Reduction

Noise reduction too produced a similar output to MATLAB prototypes. Since it was using a division by shifting it was expected to have some rounding errors. However these errors are not severe enough to be seen by an observer. Its RMS error compared to MATLAB prototype was 2.6×10^{-3}. Figure 7.7 demonstrates the result of this implementation of spatial filtering and its comparison to MATLAB prototype.

Figure 7.7 Noise reduction method results: a) Original image, b) MATLAB prototype, c) Output of the device.

The noise reduction circuitry works on the same kernel structure that defective pixel correction does and it is limited with 20 MHz similar to the defective pixel correction circuitry.

7.2.2.4 Pseudo-Coloring

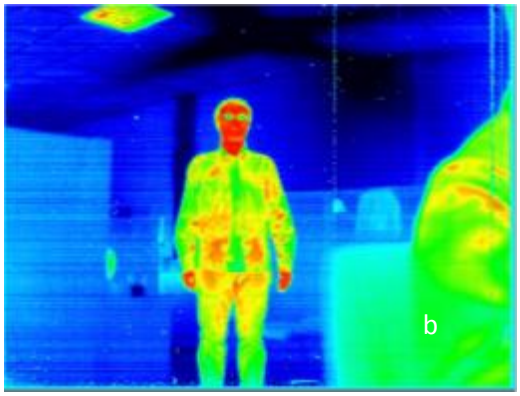
The look up table implementation for pseudo color method produced the same results as the MATLAB prototype since the look up table contents are generated with MATLAB. And the results of generic implementation were identical.

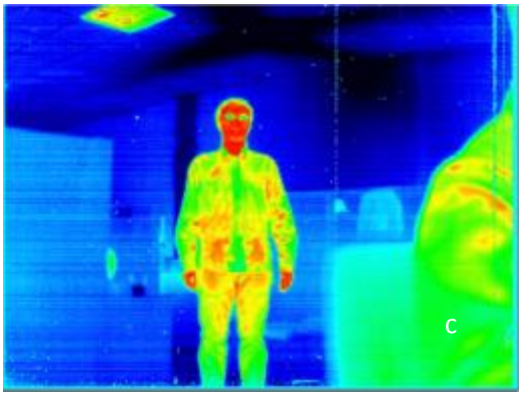
Xilinx claims that Block RAMs are capable at working with frequencies up to 333 MHz [24]. In simulation also fixed pseudo color module worked without any problems up to 125 MHz.

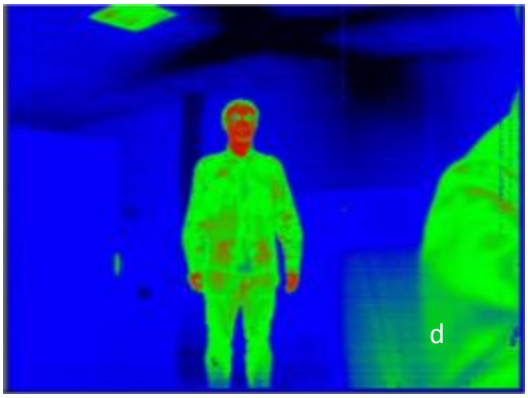
Generic implementation on the other hand, is again limited with the frequency bottleneck of divider module. For this reason it is limited with its maximum frequency of 3.3 MHz.

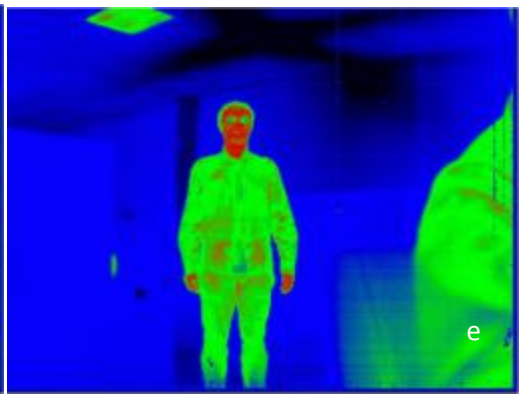
Figure 7.8 provides the results of pseudo-coloring and their comparisons to MATLAB prototypes.


a

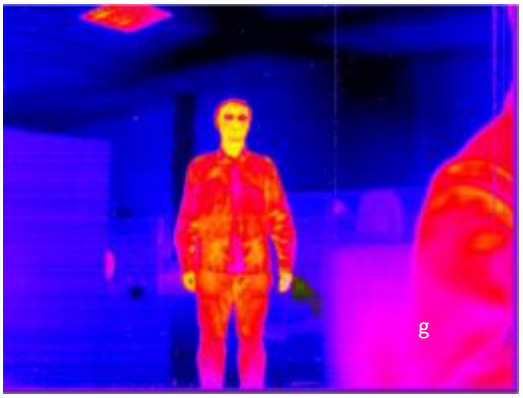

b


c


d


e


f


g

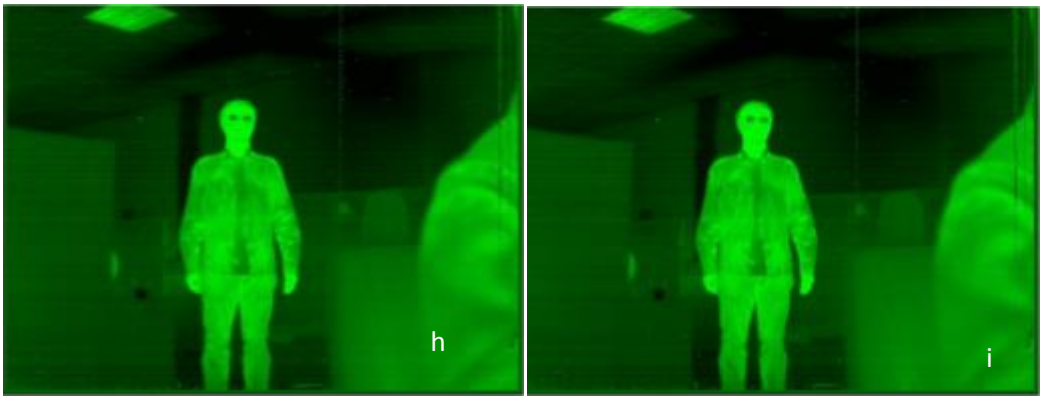


Figure 7.8 Results of pseudo coloring: a) Original image, b) Rainbow scale linear palette MATLAB prototype, c) Output of the device, d) Rainbow scale sinusoidal palette MATLAB prototype, e) Output of the device, f) Hot metal scale linear palette MATLAB prototype, g) Output of the device, h) Direct green mapping MATLAB prototype, i) Output of the device.

In addition to previously mentioned specifications the design is completely generic in terms of bit length. All the modules reconfigure themselves with the input of bit length as a parameter during the synthesis time. Furthermore the modules which have internal memory can adjust their size of memory according to the same input. The circuitries are divided into modules and operational blocks in terms of their functions. These properties make the implementation reusable in many different conditions with different requirements.

Table 7.1 provides the errors of the implemented method compared to MATLAB prototypes. This can be used as a good measure for correct implementation of the algorithms. Table 7.2 summarizes clock frequencies of the implemented methods.

Table 7.1 Comparison of the implementation results with MATLAB prototypes.

Method	RMS Error($x10^{-3}$)
Defective Pixel Correction	0.0
Contrast Streching	0.2
Histogram Equalization	0.6
Plateau Equalization (Wang et. al.)	23.6
Plateau Equalization (Lai et. al.)	2155.4
Pseudo Coloring (Rainbow Scale)	0.0
Noise Reduction (Spatial Filtering)	2.6

Table 7.2 Operating frequencies of the implementations.

Method	Frequency(MHz)
Defective Pixel Correction	20.0
Contrast Stretching	3.3
Histogram Equalization	3.3
Plateau Equalization (Wang et. al.)	3.3
Plateau Equalization (Lai et. al.)	3.3
Pseudo Coloring	125.0 / 3.3
Noise Reduction (Spatial Filtering)	20.0

CHAPTER 8

DISCUSSIONS AND FUTURE WORK

Within the scope of this, thesis camera electronics for properly operating the detector arrays and image enhancement algorithms were implemented. The requirements for IR detector arrays were discussed, and how they are provided was presented. Moreover, methods for defective pixel correction, contrast enhancement, noise reduction, and pseudo coloring were presented; their implementations were discussed, then the results of these implementations were evaluated. In this chapter, possible improvements for these methods are discussed.

Throughout the work of thesis, although the design criteria are met, some shortcomings of both algorithms and implementations were observed. Some of these shortcomings are tolerated due to area and performance tradeoffs. Some of them can be improved with additional input or components.

One shortcoming that was observed on defective pixel correction algorithms is that when there are clusters of defective pixels on the imager, median filter with a small kernel cannot come up with an estimation that is an effective pixel value. When a large kernel is used on the other hand, it filters out the important details of the image.

To get around this issue the author of this thesis developed and implemented a software algorithm. The algorithm uses a memory of defective pixels which is calculated earlier in the software. In the algorithm for each defective pixel the software counts the number of effective pixels in the 3 by 3 neighborhood. If there is at least one effective pixel the algorithm takes their median as the value for the output. If all the pixels in the kernel are defective the algorithm enlarges the kernel and counts the number of

effective pixels in the new kernel. This enlargement of kernel continues until at least one effective pixel is included in the kernel or kernel reaches a predetermined maximum value. This algorithm can be implemented in a hardware both fast and small, however in order to do this defective pixels in the imager must be know beforehand. Figure 8.1 illustrates this algorithm, and Figure 8.2 provides an example of its result.

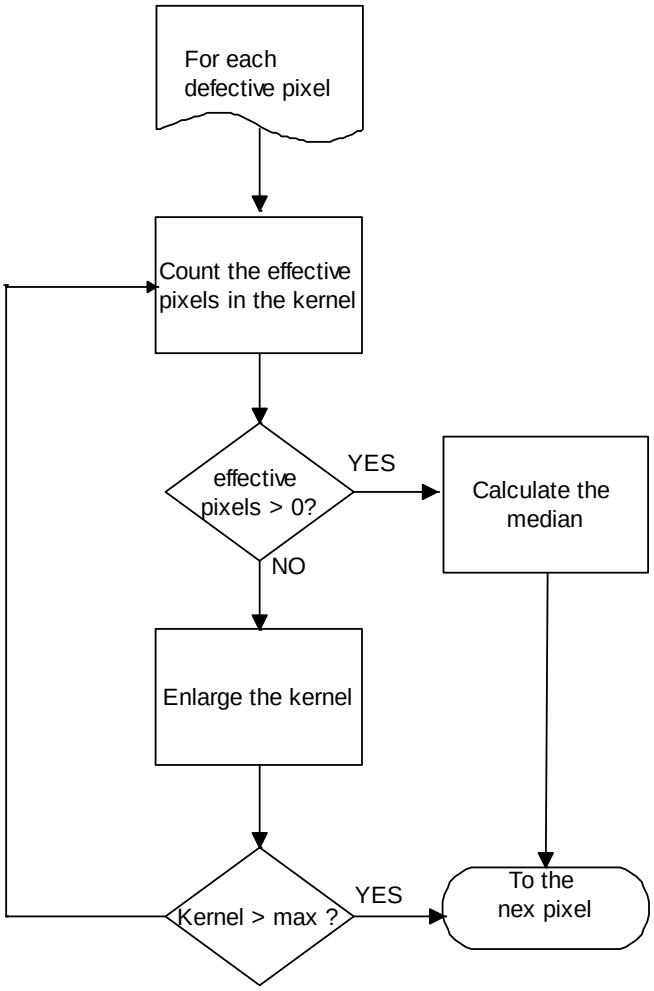


Figure 8.1 Improved defective pixel correction algorithm.

Figure 8.2 a) Raw image, b) Image result of improved defective pixel correction algorithm.

Another improvement to be made is on contrast stretching method. Author of this thesis observed that this algorithm is quite sensitive to noise since a small noise on the device would change the gain more than necessary. To get around this a percentage of the pixels can be excluded from the calculation of maxima and minima. This can be done if the maxima and minima are extracted from the histogram rather than detecting by simple maxima and minima detectors. This property could be added to the current contrast enhancement module since there is already a histogram extraction sub module present in it. Of course this would require a larger FPGA.

The author also observed that wavelet based noise reduction algorithms have better noise reduction, and less detail is lost with these methods. However these

methods require a considerable amount of memory. In a project with larger FPGAs they would increase noise reduction performance.

Implementing them on a computer is an alternative to the FPGA implementation of wavelet based methods. Computers don't have very restricted memories like FPGAs however sequential implementation of wavelet based methods may be too slow for real-time operation. Another alternative is using another specialized hardware, GPU. GPUs are commercially available and have a considerable amount of parallel processing power. [27] suggested and provided a method for implementing wavelet thresholding on a GPU. The method uses three color channels and one alpha channel for average and detail images. After that it thresholds the detail channels and combines them back into a denoised image. Figure 8.3 illustrates the implementation of multi-level wavelet thresholding on GPU.

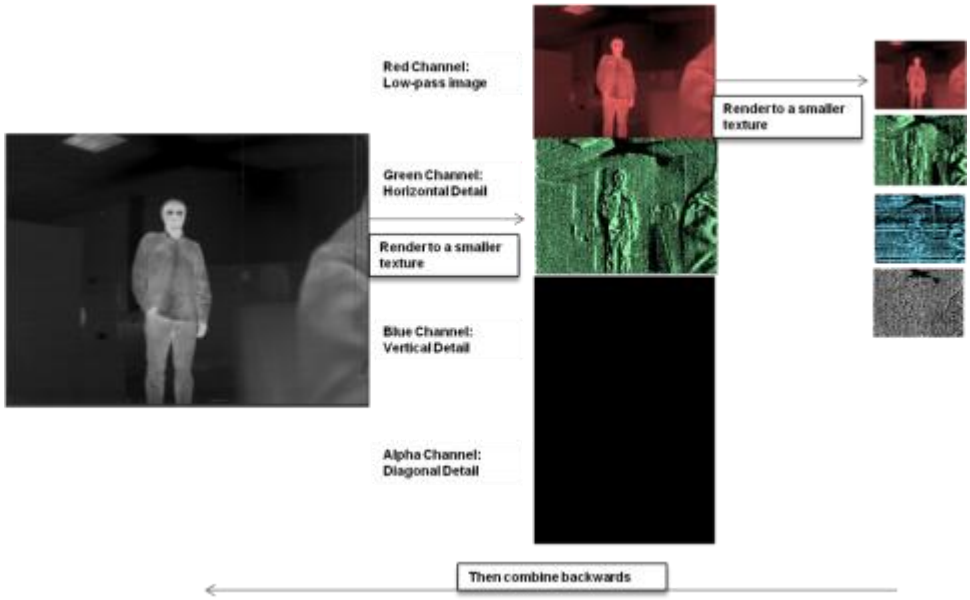


Figure 8.3 Wavelet based denoising method on GPU

The author this thesis also tried this approach as an alternative to hardware implementation. The method used by the author uses wavelet analysis to separate the average and the detail images and renders them into a smaller texture which has half the sizes of the initial image. It, then repeats this analysis step as many times as necessary, thresholds the detail images, and then combines them back into a denoised image.

However this method fails to provide a correct image. The rendering the image to a small texture and then rendering it back to a larger texture causes losses in the shape geometry of the image. This property makes this method not suitable for this application. Figure 8.4 demonstrates this error.

Figure 8.4 GPU implementation of Wavelet processing, and loss of shape geometry a) Original image, b)Re-rendered image

The largest factor affecting the performance of the implemented modules in terms of clocking frequency is the bottleneck caused by the pipelined divider. The source of the bottleneck is the long asynchronous path of shift, subtract, check-if-negative operation. This bottleneck can be overcome by retiming. But this will come with the price of using more FPGA resources. This trade off can be done if there are more free resources available on the FPGA.

Also, one can improve both the size and the clocking performance of the pipelined divider by using well known algorithms like Radix - 4 division. However for some of the generic implementation would be sacrificed with this decision.

Similarly by compromising from generic implementation, another bottleneck in terms of size can be overcome. Algorithms for smaller and relatively faster sorting [28-30] exist in the literature. One can implement such an algorithm when necessary.

Last but not least, there are some algorithms that use a considerable amount of memory only for compensating the delays on the longest path. For less memory consumption these paths could be clock with a secondary and higher frequency clock signal.

REFERENCES

[1] R. A. Wood, “Uncooled Thermal Imaging with Monolithic Silicon Focal Arrays,” *Infrared Technology XIX*, SPIE, Vol. 2020, pp. 322-329, 1993.

[2] D. S. Tezcan, S. Eminoğlu, and T. Akın, “A Low Cost Uncooled Infrared Microbolometer Detector in Standard CMOS Technology,” *IEEE Transactions on Electron Devices*, Vol. 50, No. 2, pp. 494-502, 2003.

[3] E. Alpman, “Development of Low-Cost Uncooled Infrared Detector Arrays in Standard CMOS and SOI-CMOS Processes,” *M.S. Thesis*, Department of Electrical and Electronics Engineering, Middle East Technical University, 2005.

[4] K. S. Demirci, “A Low-Cost 16x16 Uncooled Infrared Detector FPA Using Standard CMOS and Wafer Level MEMS Processes,” *M.S. Thesis*, Department of Electrical and Electronics Engineering, Middle East Technical University, 2005.

[5] D. S. Tezcan, S. Eminoğlu, O. Akar, and T. Akın, “An Uncooled Microbolometer Infrared Focal Plane Array in Standard CMOS,” *Proceedings of SPIE*, Vol. 4288, pp. 112-121, 2001.

[6] S. Eminoğlu, M. Y. Tanrıkulu, and T. Akın, “Low-Cost 64x64 Uncooled Infrared Detector Arrays in Standard CMOS,” *The 12th Int. Conf. on Solid-State Sensors and Actuators (TRANSDUCERS’03),* pp. 316-319, 2003.

[7] S. Eminoğlu, “Uncooled Infrared Focal Plane Arrays with Integrated Readout Circuitry Using MEMS and Standard CMOS Technologies,” *Ph. D. Dissertation*, Department of Electrical and Electronics Engineering, Middle East Technical University, 2003.

[8] S. Eminoğlu, M. Y. Tanrıkulu, and T. Akın, “Low-Cost Uncooled Infrared Detector Arrays in Standard CMOS,” *Proceedings of SPIE*, Vol. 5074, pp. 425-436, 2003.

[9] *2N50 Datasheet*, METU MEMS Research and Applications Center, 2010.

[10] D. Akçören, “A Low-Cost Uncooled Infrared Detector Array and Its Camera Electronics,” *M.S. Thesis*, Department of Electrical and Electronics Engineering, Middle East Technical University, 2011.

[11] J. Dudas, C Jung, G. H. Chapman, Z. Koren, and I. Koren, “Robust Detection of Defects in Imaging Arrays,” *Proceedings of SPIE-IS&T Electronic Imaging*, SPIE Vol. 6059, 2006.

[12] S. Perri, M. Lanuzza, P. Corsonello, and G. Cucorullo,” SIMD 2-D Convolver for Fast FPGA-based Image and Video Processors,” *MAPLD*, 2003.

[13] V. E. Vickers, “Plateau Equalization Algorithm for Display of High Quality Infrared Imagery,” *Optical Engineering,* Vol. 35, No. 7, 1921 – 1926, 1996.

[14] B. Wang, S. Liu, Q. Li, and H. Zhou, “A Real-Time Contrast Enhancement Algorithm for Infrared Images Based on Plateau Histogram,” *Infrared Physics & Technology,* Vol. 48, pp. 77-82, 2006.

[15] C. H. Ooi, N. S. P. Kong, and H. Ibrahim, “Bi-Histogram Equalization with a Plateau Limit for Digital Image Enhancement,” *IEEE Transactions on Consumer Electronics*, Vol. 55, No. 4, 2009.

[16] R. Lai, Y. Yang, B. Wang, and H. Zhou, “A Quantitative Measure Based Infrared Image Enhancement Algorithm Using Plateau Histogram,” *Optics Communications*, Vol. 283, pp. 4283-4288, 2010.

[17] S.M. Pizer, E.P. Amburn, J.D. Austin, R. Cromartie, A. Geselowitz, T. Greer, B.M. ter Haar Romeny, J.B. Zimmerman, and K. Zuiderveld, “Adaptive Histogram Equalization and its Variations,” *Computer Vision, Graphics and Image Processing*, Vol. 39, pp. 355-368, 1987.

[18] A. M. Reza, “Realization of the Contrast Limited Adaptive Histogram Equalization for Real-Time Image Enhancement,” *Journal of VLSI Signal Processing*, Vol. 38, pp. 35-44, 2004.

[19] X. Bai, F. Zhou, and B. Xue, “Infrared Image Enhancement Through Contrast Enhancement by Using Multiscale New Top-Hat Transform,” *Infrared Physics & Technology*, 2011.

[20] M. Nilsson, M. Dahl, and I.Claesson, “Gray-Scale Image Enhancement Using The-SMQT,” *IEEE International Conference on Image Processing*, 2005.

[21] X.Li, G. Nt, Y. Cut, T. Pu, and Y. Zhong, “Real Time Histogram Equalization Using FPGA,” *Proceedings of SPIE*, Vol. 3561 , pp. 293-299, 1998.

[22] D. L. Donoho, “De-Noising By Soft Thresholding,” *IEEE Transactions on Information Theory*, Vol. 41, No. 3, pp. 613-627, 1995.

[23] L. Ming, Z. Feiyu, and L. Qinshen, “Application of Infrared Image Noise Reduction by Processing of Wavelet Transform Threshold” *Infrared Components and Their Applications, Proceedings of the SPIE*, Vol. 5640, pp. 293-296, 2005.

[24] D. Wippig, B. Klauer, and H. C. Zeidler, “Denoising of Infrared Images by Wavelet Thresholding,” *Advances in Computer, Information, and Systems Sciences, and Engineering,* Springer, pp 103-108, 2006.

[25] R. C. Gonzales, and R. E. Woods, *Digital Image Processing*, Pearson, 3rd Edition, 2008.

[26] *DS099: FPGA Family Datasheet*, Xilinx Inc., 2009.

[27] E. A. R. Nielsen, “Real Time Wavelet Filtering on the GPU,” *M.S. Thesis*, Norwegian University of Science and Technology, 2007

[28] G. L. Bates, and S. Nooshabadi, “FPGA Implementation of a Median Filter,” *Proceedings of IEEE Speech and Image Technologies for Computing and Telecommunication*, Vol. 2, pp. 437-440, 1997.

[29] S. A. Fahmy, P. Y. K. Cheung, and W. Luk, “Novel FPGA Based Implementation of Median and Weighted Median for Image Processing,” *IEEE International Conference on Field Programmable Logic and Applications*, pp. 142-147, 2005.

[30] K. Ratnayke, and A. Amer, “An FPGA Architecture of Stable-Sorting on a Large Data Volume: Application to Video Signals,” *CISS '07. 41st Annual Conference on Information Sciences and Systems*, pp .431-436, 2007.

[31] J. F. Wakerly, *Digital Design: Principles & Practices*, Prentice Hall, 3rd Edition, 2001.