

FPGA-Based Image Change Detection Framework

by

MBIGANYI TAPELA

Reg. No: 16100806

BSc (Computer Engineering Systems) (Drexel University),

Department of Computer and Information Systems,

Faculty of Science

Botswana International University of Science and Technology

Mbiganyi.tapela@studentmail.buist.ac.bw, Phone Number (+267)71557535

A Thesis Submitted to the Faculty of Science in Partial Fulfillment of the Requirements
for the Award of the Degree of Master of Science of BIUST

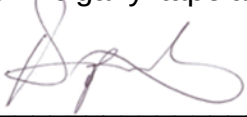
Supervisors: Dimane MPOELENG, Ph.D. & Hlomani. HLOMAN, Ph.D.

Supervisor(s): Dimane MPOELENG, Ph.D.

Department of Computer Science and Information Systems

Faculty of Science, BIUST

E-mail Address: mbiganyi.tapela@studentmail.ac.bw, Phone Number: (+267) 76 435 358

Signature:  Date: 22 October 2020

Hlomani. HLOMAN, Ph.D.

Department of Computer Science and Information Systems

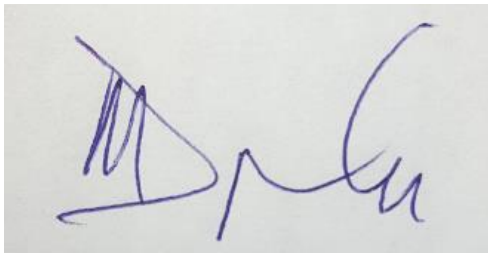
Faculty of Science, BIUST

E-mail Address: mbiganyi.tapela@studentmail.ac.bw, Phone Number: (+267) 76 435 358

Signature:  Date: 22 October 2020

DECLARATION AND COPYRIGHT

I, Mbiganyi Tapela, declare that this dissertation/thesis is my own original work and that it has not been presented and will not be presented to any other university for a similar or any other degree award.



Signature

This dissertation/thesis is copyright material protected under the Berne Convention, the Copyright Act of 1999 and other international and national enactments, in that behalf, on intellectual property. It must not be reproduced by any means, in full or in part, except for short extracts in fair dealing; for researcher private study, critical scholarly review or discourse with an acknowledgement, without the written permission of the office of the Provost, on behalf of both the author and the BIUST.

CERTIFICATION

The undersigned certifies that he/she has read and hereby recommends for acceptance by the Faculty of Science thesis titled:

FPGA-based Image Change Detection Framework, in fulfilment of the requirements for the degree of Master of Science in Computer Science of the BIUST.



.....

Dr. D. MPOELENG

(Supervisor)



.....

Dr. H. HLOMANI

(Co-Supervisor)

Date: 22 October 2020

Date: 22 October 2020

ACKNOWLEDGEMENTS

I would like to express my sincere appreciation and gratitude to my supervisors; Dr. Dimane Mpoeleng and Dr. Hlomani Hlomani for their guidance, encouragement and contributions. Their offices were always open whenever I needed help or had a question about my research. I would like to thank them for the advice and support they gave me throughout my Master's program.

My special appreciation also goes to the staff of Computer Science and Information Systems department for their brilliant comments, suggestions and advice. Lastly, I am thankful to all those who have supported me directly or indirectly during my research.

Abstract

This research presents a new approach in modeling and implementing a framework for linear image change detection using Field Programmable Gate Arrays (FPGA) based implementations. Detecting changes between digital images of the same location or region of interest have attracted huge interest in research due to its wide range of applications such as medical diagnosis, astronomy and forensics. In recent years, the steady increase in the use of hardware implementation in image processing, especially the FPGA architecture that has proven to be superior in performance and cost-efficiency when compared to the conventional computer systems. The problem addressed by this research is the lack of an FPGA based generic and standardized change detection framework that executes image change detection at a low computational cost, using low processing resources like memory and power consumption.

The framework proposed in this research work is focused on the image differencing, which is an algebra change detection technique. Image edge detection, image enhancement and restoration methods were employed in the framework implementations using MATLAB/Simulink and the Xilinx system generator. The success of the framework implementation proved that the objectives of the thesis were achieved and has been verified by the experimental results. While the implementation was successful in producing expected results, there is a need for more experiments to explore different filtering algorithms that could improve the results. Statistical forms of data interpretation such as Machine Learning algorithms could improve complex image difference results analysis.

Table of Contents

Abstract.....	vi
List of Synonyms	xi
List of Tables	xii
List of Figures.....	xii
Chapter 1: Introduction	1
1.0 Overview	1
1.1 Digital Image Processing	1
1.2 Change Detection and Field Programmable Gate Arrays	2
1.3 Frameworks for Change detection	5
1.4 Generic and Standardized Framework.....	5
1.5 Order of Image Enhancement and Restoration Processes	6
1.6 Use of FPGA in Image Processing	7
1.7 Problem Statement.....	8
1.8 Objectives.....	9
1.8.1 General Objective	9
1.8.2 Specific Objectives	9
1.9 Research Questions	9
1.10 Significance of the Study.....	10
1.11 Organization of the Thesis.....	11
Chapter 2: Literature Review	12
2.0 Overview	12
2.1 Background.....	12

2.1.1 Field Programmable Gate Arrays	12
2.1.2 Change Detection	13
2.1.3 Image Pre-processing	14
2.1.4 Image Restoration and Enhancement	15
2.2 Related Work.....	15
2.2.0 Overview	15
2.2.1 Edge Detection	15
2.2.2 FPGA-based Frameworks.....	20
2.2.3 Chapter Summary	28
Chapter 3: Framework Modeling and Implementation.....	29
3.0 Overview	29
3.1 Proposed FPGA-based Change Detection Framework	29
3.2 Components of the Proposed Framework	31
3.2.1 Image Acquisition.....	33
3.3 Image Enhancement and Restoration	34
3.3.1 Digital Image Denoising.....	37
3.3.2 Image Deblurring.....	38
3.3.3 Image Sharpening	42
3.3.4 Image Registration.....	43
3.4 Edge Detection	44
3.4.1 Vertical Edge Detection.....	47
3.4.2 Horizontal Edge Detection	48
3.4.3 Binary Conversion.....	48
3.5 Image Change Detection.....	49
3.5.1 Image Subtraction Using Exclusive-OR Operator	50

3.5.2 Exclusive-or (XOR) Operation.....	51
3.5.3 Implementations of Boolean Equations.....	53
3.5.4 Contrast Stretching	54
3.5.5 Min-Max Linear Contrast Stretching	55
3.6 Design and Modeling of the Framework.....	56
3.6.1 System Implementation	60
3.6.2 MATLAB/Simulink.....	61
3.6.3 Xilinx System Generator (XSG).....	61
3.6.4 Pre-processing for FPGA Implementation Using XSG.....	62
3.6.5 Post Processing for Image Enhancement Processes	63
3.6.6 Framework Design and Experimental Setup	63
3.7 Framework Implementation	67
3.7.0 Overview	67
3.7.1 Implementation of the Image Change Detection Framework	67
3.7.2 FPGA Implementation of Image Denoising	70
3.7.3 FPGA Implementation of Image Deblurring.....	71
3.7.4 FPGA Implementation of Image Sharpening Using XSG.....	73
3.7.5 FPGA Implementation of the Sobel Edge Detection Using XSG	75
3.7.6 Implementation of Change Detection Using ISE	79
3.7.7 FPGA Implementation of Image Change Detection Using XSG	81
3.7.8 FPGA Implementation of Contrast Stretching	84
Chapter 4: Discussion and Analysis of Results	85
4.0 Overview	85
4.1 Results from Components of the Proposed Framework.....	85
4.1.0 Overview	85

4.1.1 Results for Image Denoising	85
4.1.2 Results for Image Deblurring	89
4.1.3 Results of Image Sharpening Process.....	91
4.1.4 Results for the Sobel Edge Detection	96
4.2 Results for Image Subtraction Simulation Using the ISE	100
4.2.1 Output Waveform for Input Signal pair ‘0 0’	102
4.2.2 Output Waveform for Input Signal pair ‘0 1’	103
4.2.3 Output Waveform for Input Signal pair ‘1 0’	103
4.2.4 Output Waveform for Input Signal pair ‘1 1’	104
4.3 Results of the Change Detection Framework	106
4.4 Results for Contrast Stretching	107
4.5 Framework Evaluation	108
Chapter 5: Conclusion.....	115
5.1 Conclusion.....	115
5.2 Future Work	116
Reference	118
APPENDIX: MATLAB M-Files and VHDL Codes	125

List of Synonyms

FPGA	Field Programmable Gate Arrays
GPU	Graphical Processing Unit
HDL	Hardware Description Language
HPC	High Performance Computing
I/O	Input/Output
IOB	Input/Output Block
ISE	Integrated Software Environment
RAM	Random Access Memory
RS	Remote Sense
VHSIC	Very High-Speed Integrated Circuit
XSG	Xilinx system generator

List of Tables

TABLE 2. 1 LITERATURE REVIEW FOR CHANGE DETECTION FRAMEWORKS	ERROR! BOOKMARK NOT DEFINED.
TABLE 3. 1 TRUTH TABLE FOR THE EXCLUSIVE-OR OPERATOR.....	51

List of Figures

FIGURE 1. 1 PROPOSED FUNDAMENTAL STEPS FOR IMAGE CHANGE DETECTION	4
FIGURE 3. 1 THE PROPOSED FPGA-BASED CHANGE DETECTION FRAMEWORK.....	31
FIGURE 3. 2 KEY STAGES OF THE PROPOSED GENERIC CHANGE DETECTION FRAMEWORK	33
FIGURE 3. 3 KEY STEPS OF IMAGE ENHANCEMENT AND RESTORATION	37
FIGURE 3. 4 IMAGE DEBLURRING TECHNIQUES.....	39
FIGURE 3. 5 DEGRADATION MODEL OF IMAGE.....	40
FIGURE 3. 6 IMAGE SHARPENING USING UNSHARP MASKING	43
FIGURE 3. 7 ORIGINAL SOURCE IMAGE	47
FIGURE 3.8 IMAGE WITH VERTICAL EDGES	48
FIGURE 3.9 IMAGE HORIZONTAL EDGES.....	48
FIGURE 3.10 FLOW DIAGRAM FOR IMAGE SUBTRACTION USING XOR OPERATOR	52
FIGURE 3.11 XOR GATE IMPLEMENTED WITH FOUR NAND GATES.....	52
FIGURE 3.12 (A) INPUT IMAGE A, (B) INPUT IMAGE B (C) OUTPUT IMAGE C (DIFFERENCE IMAGE)	53
FIGURE 3.13 ORIGINAL IMAGE (LOW CONTRAST)	56
FIGURE 3.14 IMAGE WITH HIGH CONTRAST AFTER CONTRAST STRETCHING	56

FIGURE 3.15 SIMULINK/XILINX SYSTEM GENERATOR DESIGN FLOW	57
FIGURE 3.16 1 ST DATE ORIGINAL IMAGE	58
FIGURE 3.17 1 ST DATE IMAGE (GRAY SCALE)	58
FIGURE 3.18 2 ND DATE ORIGINAL IMAGE	59
FIGURE 3.19 2 ND DATE IMAGE (GRAY SCALE)	59
FIGURE 3.20 DESIGN FLOW FOR FRAMEWORK IMPLEMENTATION	60
FIGURE 3.21 IMAGE PRE-PROCESSING BLOCKS UNIT	62
FIGURE 3.22 IMAGE POST-PROCESSING BLOCKS UNIT	63
FIGURE 3.23 IMAGE ENHANCEMENT AND RESTORATION PHASE	65
FIGURE 3. 24 IMAGE CHANGE DETECTION PHASE	66
FIGURE 3.25 1ST DATE IMAGE (BLURRY AND NOISY)	68
FIGURE 3.26 2ND DATE IMAGE (BLURRY AND NOISY)	68
FIGURE 3.27 1ST DATE IMAGE (NOISY AND BLURRY)	70
FIGURE 3.28 2ND DATE IMAGE (NOISY AND BLURRY)	71
FIGURE 3.29 IMAGE NOISE REMOVAL XILINX/SIMULINK MODEL DESIGN	71
FIGURE 3.30 1ST DATE IMAGE (BLURRY, NO NOISE)	72
FIGURE 3.31 2 ND DATE IMAGE (BLURRY, NO NOISE)	72
FIGURE 3.32 IMAGE BLUR REMOVAL USING XILINX/SIMULINK MODEL DESIGN	73
FIGURE 3.33 IMAGE SHARPENING USING XILINX/SIMULINK MODEL DESIGN	73
FIGURE 3.34 1ST DATE IMAGE (BLUR FREE AND NOISE FREE)	74
FIGURE 3.35 2ND DATE IMAGE (BLUR FREE AND NOISE FREE)	74
FIGURE 3.36 1ST DATE IMAGE (SHARPENED)	75
FIGURE 3.37 2ND DATE IMAGE (SHARPENED)	76
FIGURE 3.38 IMAGE EDGE DETECTION USING SOBEL OPERATOR	78
FIGURE 3. 39 SCHEMATIC FOR IMAGE SUBTRACTION USING XOR OPERATOR	79
FIGURE 3. 40 SIMULATION OF IMAGE SUBTRACTION USING XOR OPERATOR	80

FIGURE 3. 41 1 ST DATE IMAGE (EDGES DETECTED)	81
FIGURE 3. 42 2 ND DATE IMAGE (EDGES DETECTED).....	81
FIGURE 3. 43 BINARY 1 ST DATE IMAGE.....	82
FIGURE 3. 44 BINARY 2 ND DATE IMAGE	82
FIGURE 3.45 DESIGN FLOW FOR IMAGE CHANGE DETECTION IMPLEMENTATION	83
FIGURE 3. 46 CONTRAST STRETCHING BLOCK DESIGN.....	84
FIGURE 4. 1 1 ST DATE IMAGE (NOISY & BLURRED IMAGE)	86
FIGURE 4. 2 1 ST DATE IMAGE RESTORED IMAGE (NOISE REMOVED)	87
FIGURE 4. 3 2 ND DATE IMAGE (NOISY & BLURRED IMAGE).....	87
FIGURE 4. 4 2 ND DATE IMAGE (NOISE REMOVED)	88
FIGURE 4. 5 1 ST DATE IMAGE (BLURRED).....	89
FIGURE 4. 6 1 ST DATE IMAGE (RESTORED IMAGE)	89
FIGURE 4. 7 2 ND DATE IMAGE (BLURRED)	90
FIGURE 4. 8 2 ND DATE IMAGE (RESTORED IMAGE)	90
FIGURE 4. 9 1 ST DATE IMAGE.....	92
FIGURE 4. 10 1 ST DATE IMAGE (SLIGHTLY BLURRED).....	92
FIGURE 4. 11 MASK (1 ST DATE IMAGE)	93
FIGURE 4. 12 2 ND DATE IMAGE	93
FIGURE 4. 13 2 ND DATE IMAGE (SLIGHTLY BLURRED)	94
FIGURE 4. 14 2 ND DATE IMAGE (MASK)	94
FIGURE 4. 15 SHARPENED IMAGE	95
FIGURE 4. 16 SHARPENED IMAGE	95
FIGURE 4. 17 1 ST DATE IMAGE.....	97
FIGURE 4. 18 2 ND DATE IMAGE	97
FIGURE 4. 19 1 ST DATE IMAGE (OUTPUT IMAGE WITH EDGES DETECTED)	98

FIGURE 4. 20 BINARY 1 ST DATE IMAGE.....	98
FIGURE 4. 21 2 ND DATE IMAGE (OUTPUT IMAGE WITH EDGES DETECTED).....	99
FIGURE 4. 22 BINARY 2 ND DATE IMAGE	100
FIGURE 4. 23 WAVEFORM HIGHLIGHTING THE OUTPUT AND INPUT SIGNAL PAIR '0 0'	102
FIGURE 4. 24 WAVEFORM HIGHLIGHTING THE OUTPUT AND INPUT SIGNAL PAIR '0 1'	103
FIGURE 4. 25 WAVEFORM HIGHLIGHTING THE OUTPUT AND INPUT SIGNAL PAIR '1 0'	104
FIGURE 4. 26 WAVEFORM HIGHLIGHTING THE OUTPUT AND INPUT SIGNAL PAIR '1 1'	104
FIGURE 4. 27 DIFFERENCE IMAGE (OUTPUT OF THE CHANGE DETECTION PROCESS)	106
FIGURE 4. 28 CONTRAST STRETCHING	108

Chapter 1: Introduction

1.0 Overview

The motivation for creating and implementing an FPGA-based change detection framework comes from the steady increase in the use of hardware implementation in image processing, especially the FPGA architecture that has proven to be superior in performance and cost efficiency when compared to the conventional computer systems. The lack of a generic framework for image change detection has also motivated the creation of a standardized method for implementing efficient image change detection process.

This thesis presents the research on an image change detection framework that is based on Field Programmable Gate Array (FPGA) architecture. It highlights the major topics in, change detection, FPGAs, and the related subjects in digital image processing. This chapter includes a discussion of factors that have motivated the research that are different from the status quo. The problem statement, the objectives of the research and the significance of the research are also discussed at length and lastly, the organization of this thesis is outlined with a brief description of each chapter of the thesis.

1.1 Digital Image Processing

An image is defined as a two-dimensional light intensity function $f(x, y)$, where (x, y) represents the spatial coordinates and the value of f (magnitude) at any point (x, y) is proportional to the brightness or gray levels of the image [1]. When the spatial coordinates and amplitude values are all finite (discrete values), the image is referred to as a digital image. Images are made up of picture elements or pixels, which are the smallest addressable elements of a digital image [4]. Digital images are processed by a computer using a process referred to as digital image processing and for this process to take place, an image has to be digitized both spatially and in amplitude.

The two principal application areas of digital image processing are the improvement of pictorial information for human visual interpretation, and processing of image data for storage,

transmission and representation for autonomous machine perception [4]. In order to prepare digital images for improved human interpretation and further processing and analysis, they have to be subjected to low-level processing operations where the images go through noise reduction, blur removal, color correction, image sharpening, contrast stretching and other image enhancement and restoration processing.

1.2 Change Detection and Field Programmable Gate Arrays

Detecting changes between digital images of the same location or region of interest has attracted huge interest in research due to its wide range of applications such as medical diagnosis, astronomy, forensics and background segmentation [2], [3], [4]. Change detection is a digital image processing technique that plays a vital role in image analysis and interpretation. This image processing technique involves both two-dimensional and multidimensional images. However, these processes demand a lot of computation due to their complexity and high performance. In the last decade, the world has experienced the move or shift in image resolutions from Standard Definition (SD) to High Definition (HD) [5]. This shift represents six times increase in data that needs to be processed by image devices. This trend is common in video surveillance, medical imaging, and the entertainment field in particular where the 4k resolution has become popular.

The current trend is that very large high-resolution imagery from equipment like drones, satellites, medical scanners and astronomy are becoming very useful, but the challenge has been the speed at which the high-resolution data is processed and analyzed. This trend calls for image analysis systems and algorithms that can handle this data. Digital Signal Processor (DSP) requires multiple processors or a single and very expensive high-end processor. But a single FPGA embedded device is capable of delivering exceedingly more computing power at cost-effective operation than the multiple core DSP system [6]. For this reason, the FPGA systems have continued gaining preference from researchers as a cost-effective alternative image-processing platform in high performance computing.

This thesis presents an approach for developing a generic framework for change detection based on an FPGA, a computing technology that has superior performance and high-speed parallel computing capacity [7]. By virtue of their high degree of parallelism and low energy consumption, FPGAs are especially well suited for the acceleration of compute-intensive image and signal

processing [7]. FPGA was first developed in 1984 by Ross Freeman, a co-founder of Xilinx Inc [8]. Several brands of FPGAs exist in the market and the leading companies in design of FPGAs are Xilinx Inc and Altera Corporation [9].

In recent years technology advances in image processing tends to focus on custom hardware implementation of computationally intensive algorithms to achieve high-speed performance. The computer systems taking the lead in speed are the programmable logic device embedded systems, in particular the FPGA [9]. Therefore, FPGA systems have become viable platforms for the implementation of high-performance frameworks suited for image processing applications. The field-programmable property of an FPGA allows the functionality of the circuit to be altered to suit any specific application. Unlike the traditional computer architectures such as general-purpose processors, a user can configure and reconfigure them any time after manufacturing.

The implementation of a generic framework for change detection algorithm using an FPGA architecture has not been fully explored. For this reason, the research has dwelt on real-time change detection FPGA-based framework to leverage the high performance, field re-programmability and re-usability the FPGA technology possesses. To the best of this researcher's knowledge and far as literature has been surveyed as described in Table 2.1, there is no generic real-time framework for executing change detection at a low computational cost, using low processing resources like memory and power consumption. The frameworks that exist are not fully generic, they only address change detection for a specified application or image-processing project.

In formulating the framework, this research has focused on the common and widely used algebra change detection techniques, which include image differencing, image ratioing, background subtraction, image regression, and change vector analysis. In this approach change detection framework will be formulated to encompass the algebra change detection [10]. One of the common techniques in change detection technique is image algebra, and it can be classified into two types: the band subtraction and the band ratio methods [11].

The generic FPGA-based framework is presented and illustrated in the context of common change detection methods focusing on the algebra change detection techniques, such as image differencing, image ratioing, background subtraction, image regression and vector analysis (CVA) [11]. These change detection methods were selected for the framework because they are closely

related. They possess a shared feature of picking thresholds to establish and identify the regions that have been affected. The key factor in these algorithms is identifying suitable image bands and thresholds. However, the vector analysis has an exceptional characteristic in that it can pick detailed changes that are over the fixed thresholds.

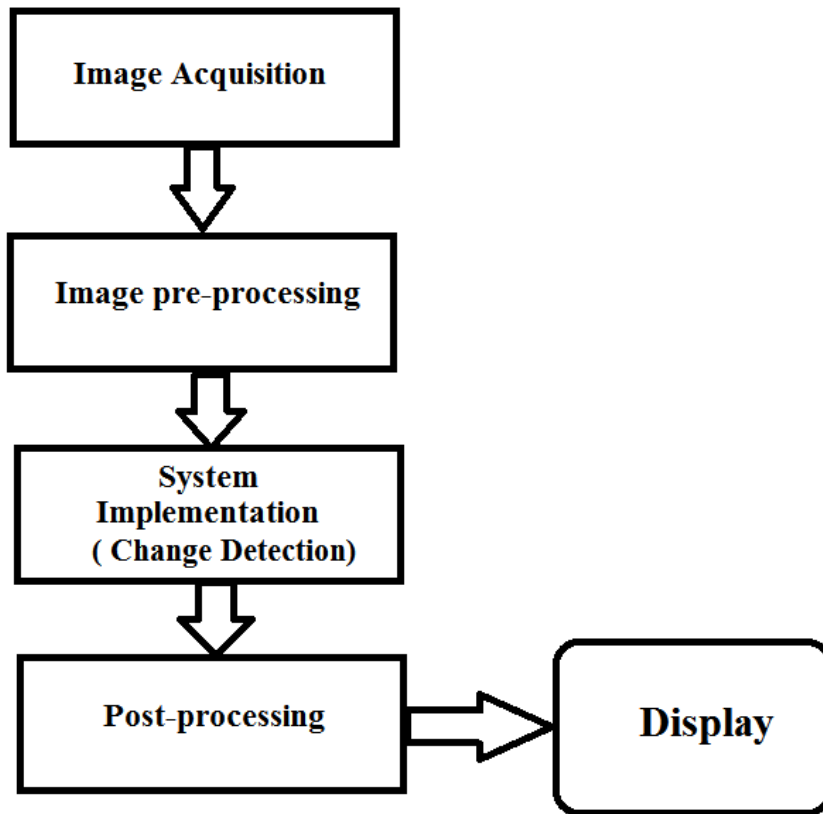


Figure 1. 1 Proposed Fundamental Steps for image Change Detection

Figure 1.1 depicts the processes of the proposed fundamental procedures or steps that are necessary in implementing image algebra change detection algorithms; image acquisition, image pre-processing, image processing and post-processing. For each of the image processing stages, there are several operations that are implemented as described later in the Chapter 3.

The final step within the design process is system implementation or image processing. This is where the change detection algorithm is mapped onto the chosen FPGA resources as defined by

the architecture. In this processing stage, specific hardware has been designed and made architecturally compatible to perform the change detection operations. Within the system implementation image manipulating algorithms are implemented and consequently the processes of detecting changes are performed.

1.3 Frameworks for Change detection

This subsection discusses factors that have motivated the research undertaken in this thesis which are different from the status quo. There exist several image change detection framework design approaches in the image processing literature, and none of them outlines a generic procedure for image change detection. The literature that substantiates the need for a generic and standardized framework for change detection is discussed in the sections that follow. The other factor that motivated this research is the sequential order of the image pre-processing operations that has a significant influence in the accuracy of the final output of the change detection process.

1.4 Generic and Standardized Framework

The literature on change detection frameworks shows that most frameworks are not generic; they are designed for specific systems. Therefore there is a need for an FPGA-based framework that can be applied to most image change detection applications

According to literature in Benkrid et al. [12], the framework's library contains optimized implementations tailored for a single FPGA embedded system, which greatly limits its use on other FPGA systems. The framework implemented by Salcic et al. [13], improves the image enhancement using FPGA but does not go further to include image change detection, it is also limited to X-ray image processing. In the framework designed by Rao et al. [14], the implementation is not scalable and is less generic in that it uses a Xilinx generator, proprietary library that is not available in other FPGA systems, hence giving a limitation for other systems. Given these limitations and constraints, the proposed framework design is more generic, closing the gaps in the previous works.

The literature in change detection and image processing frameworks show that the frameworks employ a series of image processing algorithms, image enhancement and image restoration processes [15]. These processes are implemented on various image formats for varying applications.

In the literature reviewed, there is a lack of a generic and standardized framework for image change detection. According to [16], for change detection process to yield accurate results, they ought to be a proper procedural image enhancement processes and image registration that prepare the images for change detection. These factors have motivated the design of a framework that uses an image pre-processing sequential procedure or steps for change detection. Some of the frameworks that do exist in the literature do not implement image change detection procedures but address other image processing implementations such skin detection and motion detection algorithms[17]. The pros and cons of such frameworks have been discussed in the literature review section described in Table 2.1.

1.5 Order of Image Enhancement and Restoration Processes

The sequential order of image processing and pre-processing have also motivated the proposed framework. The order of the image operations is crucial and the literature review has showed that when a carefully ordered steps are not followed, the performance of the change detection process is compromised and cannot show all the actual changes between two images. It is recommended that noise removal be applied the earliest in image enhancement since it is detrimental to subsequent image analysis. If further image enhancement is done on images that contain noise, the accuracy of the results is significantly affected [18]. According to [19], order of enhancement matters, and one of the enhancement processes that influence the accuracy of image analysis is the image sharpening. This process must be implemented after the removal of blur has been performed. When an image is sharpened before motion blur is removed, the frequencies get amplified during the sharpening process causing inaccurate results. Therefore in this thesis, a careful consideration has been taken into account when creating a design flow for the framework. The image pre-processing steps have been arranged such that the output of a

process does not in any way adversely affect process in the subsequent steps. In this way, the framework achieves optimal results.

1.6 Use of FPGA in Image Processing

As the world's demand for use images for various applications steadily increases, recent research studies indicate that demand comes with the challenge of processing this data in an efficient and less costly way. A key motivation is the use of FPGA systems in image processing implementations, which is becoming increasingly prevalent in image processing applications.

Energy efficiency and parallelism are the main features of FPGA that make it the preferred platform for performing arithmetically intensive algorithms [20]. The other factor that makes the use of an FPGA system a very useful phenomenon is the speed at which FPGA implements image processing on raw images and high-resolution images, which are normally huge in size. The processing of huge image data on FPGA systems is extremely fast compared to the speed in general purpose computer systems, executing operations instantly [21]. The FPGA computational architecture can be configured to suit a specific algorithm instead of adapting an algorithm to a fixed architecture [22]; this feature makes it possible for the whole chip to be used all the time since operations run every clock cycle. In the proposed framework, some image processing operations will be performed in parallel to harness the FPGA power of parallelism to speed up execution.

The FPGA system has unique characteristics, which include cost effectiveness in handling computationally intensive processes [22]. Image processing algorithms are computationally intensive especially when pixel-based change detection implementation is performed.

The motivation in using FPGA-based image processing in this research also comes from pipelining. Pipelining is a technique for improving system performance and throughput [23]. This FPGA structure leverages the inherent parallelism in instructions that are divided into subtasks. These instructions are executed at the same time thereby increasing the throughput [24]. In image processing, some of the operations for image scaling and transformation can be executed simultaneously using pipelining [25]. A computationally heavy section of a program is identified

and reprogrammed in a pipelined and parallel fashion, a technique that is either impossible or not efficient in other computer system

An FPGA system outperforms the multi-core and CPUs in image processing and its efficiency is 501 higher. An FPGA system is also the most flexible amongst the three systems [26]. Though GPUs are comparatively fast, the major disadvantage of GPUs is the cost, which is very high, compared to the FPGA system. Therefore the FPGA has become the preferred system because of its advantage of re-programmability, and cost effectiveness [26].

The framework proposed in this thesis is characterized with image enhancement and image processing algorithms that require very high computational power and the FPGA system is the system deemed to be better suited to tackle the challenge of execution time.

When using multi-core CPUs to process huge image data such as very high-resolution images, satellite images and raw format images, the execution time increases due to the low processing speed; on the other hand the effect of size and format of the image data in FPGA implementation is negligible, because images are converted to binary rendering image format constraint irrelevant.

1.7 Problem Statement

Image change detection is compute-intensive and takes a long time to execute especially when pixel-based algorithms are performed. The problem addressed by this research is the lack of an FPGA based generic and standardized change detection framework that executes image change detection at a low computational cost, using low processing resources like memory and power consumption. An FPGA based image change detection framework is proposed, where different configurations are tailored for specific applications, resulting in fast computational speed, low memory and power consumption.

1.8 Objectives

1.8.1 General Objective

To develop a generic and standardized image change detection framework that leverages the FPGA technology to perform image processing at fast speed and low computational cost.

1.8.2 Specific Objectives

- To develop suitable procedures for image enhancement and restoration algorithms required for effective image change detection
- To implement and deploy an FPGA based framework for an image change detection
- To test and evaluate the FPGA-based image change detection framework

1.9 Research Questions

In addressing the objectives, the following questions were answered by this study:

- (i) What FPGA implementation is suitable for a change detection framework?
- (ii) Which are the most applicable image processing algorithms and hardware implementations that can be well matched on an FPGA system for image change detection?
- (iii) What are the common change detection operations?
- (iv) How can FPGA implementations be generalized for change detection?
- (v) How can a generic framework for different change detection implementations be constructed?
- (vi) How effective are the various types of image enhancement and restoration in improving the quality of image change detection?
- (vii) What effect does the use of sequential order of image processing and pre-

processing have on the difference image?

- (viii) When image change detection is implemented, how do results from enhanced images and from those that are not enhanced compare?

1.10 Significance of the Study

The implementation of a generic and systematic change detection using an FPGA-based framework has not been fully researched. Performing complex image processing algorithms is computationally intensive and implementing this process on a central processing unit (CPU) or a graphical processing unit (GPU) based system takes a long time and consumes lot of power and memory. In comparison, the implementation process in FPGA-based hardware architecture is superior because of its features of parallelism, shared memory and low power consumption. The parallel processing capabilities make it possible for all processes to be executed or run at the same time in parallel, hence achieving short execution time. So this research will contribute in opening more opportunities for further research in field programmable gate array FPGA-based generic frameworks.

1.11 Organization of the Thesis

The organization of this thesis is as follows:

- Chapter 1 provides the introduction of the Image change detection, field programmable gate arrays (FPGA) and related technologies. This chapter also covers the problem statement, the research motivation and the objectives of the research.
- Chapter 2 presents the theoretical background, detailed literature survey and it also reviews different techniques that are used for image enhancement and restoration, edge detection, change detection, FPGA and hardware-based image processing related frameworks. The other subjects this chapter covers are image subtraction, MathWorks/Xilinx development and testing environment.
- Chapter 3 discusses the framework modeling and implementation in great detail. The chapter provides the framework design model and stages making up the framework. The topics covered in this chapter are; image enhancement and restoration steps that include image acquisition where images are acquired using digital equipment. The image filtering techniques include denoising, deblurring and the sharpening algorithms. Other subjects include the edge detection/extraction using Sobel edge detection algorithm, and contrast stretch process for improved image contrast and quality. The last topic on image processing discussed in this chapter is the change detection algorithm, which covers image subtraction using the exclusive-OR operation. The framework modeling and implementation is covered here, and it is all based on the MathWorks/Xilinx integrated environment in which MATLAB/Simulink and Xilinx system generator is used for all the image processing algorithms implementations.
- Results and discussions of the framework implementation are provided in Chapter 4.
- Chapter 5 closes the thesis with overall conclusion derived from the current work, along with an outlook on the future work.

Chapter 2: Literature Review

2.0 Overview

Chapter 2 provides a review and discussions of performance, relative merits and limitations of the research fields related to FPGA-based change detection framework design approaches. The relevant fields that are surveyed include; image enhancement and restoration techniques, field programmable logic devices, change detection and edge detection algorithms. This review also evaluates the different frameworks and design approach techniques used in this research subject with the intent of determining if standardized methodologies do exist.

Image change detection is a subject with a wide range of image processing techniques that can be classified into several categories and sub-categories. There are many change detection framework design approaches for these change detection categories and their sub divisions that are provided in the literature.

The scope of this review is limited to FPGA-based change detection frameworks and the review will not explore image-processing frameworks that are based on other computing architectures such as the implementation of multi-core CPU and graphic processing unit (GPU) based change detection frameworks.

2.1 Background

2.1.1 Field Programmable Gate Arrays

An FPGA is a high-capacity field-programmable device (FPD), also referred to as programmable logic device (PLD). A FPGA is a circuit design that can be reconfigured by a 609 customer in the field for a specific function after manufacturing [28]. The other two field-programmable architectures that are commercially available are the simple programmable logic devices (SPLDs)

and complex programmable logic devices (CPLDs) [27]. The digital designs of recent years have largely adopted the PLD architecture as the core technology used to build electronic systems. This has led to high-density devices dominating most of the digital designs in the market. However, in most cases the cost of development or non-recurring engineering (NRE) and the manufacturing costs of PLD based designs are lengthy and relatively very expensive especially for low-volumes and prototyping purposes [28]. These high costs of production have given rise to FPGA based designs due to their lower production costs and their suitability for prototyping. The FPGAs have a wider potential applications than other programmable logic devices due their flexibility of being field programmable, giving users the capability to tailor processors to meet their own specific needs.

2.1.2 Change Detection

Despite the numerous advances made in previous decades toward developing systems that could handle image change detection in real-time, programmable logic devices remain the primary focus for most real-time systems. A design of a generic change detection framework that combines a linear type of change detection technique and its hardware implementation is rarely achieved, this is because algorithms development typically focuses on achieving functional correctness while implementation focuses on achieving maximum efficiency or minimal computational resources. It is not enough to port an algorithm from software onto FPGA architecture. The algorithm and the computational architecture implementation must be well matched though there is no unique solution [29].

Even though programmable logic devices especially FPGA hardware architecture is not a typical implementation in such frameworks, there has been tremendous research going on in this field. FPGA embedded systems have emerged as important systems for applications such as digital image processing and image analysis due to their flexibility and high-performance capabilities in comparison with other systems such as general-purpose computers and Application Specific Integrated Circuits (ASICs).

Image change detection methods can be classified in two distinct categories: supervised and unsupervised [30], or pixel-based and object-based change detection depending on the applications [31]. Image change detection techniques are implemented using various hardware/software applications and high-performance computing, which has been steadily

attaining popularity in recent years. The FPGA based architecture is one of the high-performance computing technologies that is used in image processing implementations. FPGA is attracting a lot of research and it is rapidly gaining great favor in many applications due to its many advantages over other systems.

The implementation of change detection involves other image processing processes especially image pre-processing techniques and these techniques can be classified into enhancement and restoration processes. The principal purpose of image pre-processing is to improve the quality of the image for improved human visual interpretation and to recover images from the degraded status. Image enhancement and image restoration is a very wide subject and it covers very many image filtering algorithms such as denoising, deblurring, contrast stretching, image sharpening, etc. Every one of the image filtering algorithms can be classified into several image-processing techniques that find use in various. In this research the image pre-processing techniques and other image processing algorithms were all implemented as part of the FPGA-based framework for change detection.

2.1.3 Image Pre-processing

Image pre-processing can be defined as a series of image operations at a lower level of abstraction that aims at improving image data for further processing [32]. Image pre-processing dramatically improves results of image analysis and the image quality for feature extraction. The process manipulates and converts digital images, which are acquired from various imaging modalities as raw data into relevant data for further image processing and analysis. In this thesis, the focus is on two-dimensional (2D) image processing techniques such as image noise removal, image blur removal, image sharpening and image registration.

Image pre-processing enhances the images by removing impurities or unwanted signals in images and makes the output image suitable for further image processing or analysis. A raw digital image that has been affected by motion blur or noise does not give true representation of the original image of an object due to the undesired impurities such as motion blur or noise, [33]. Undesired elements must be removed in order to obtain an image that closely resembles the ideal original image. There are two image pre-processing categories: image restoration and image enhancement.

2.1.4 Image Restoration and Enhancement

Image restoration is a sub field of digital image processing that attempts to obtain a close estimate of the original image from a corrupted image [34]. Image restoration algorithms are based on models for the degrading process of the noisy blurred image and for the original image [35]. The main goal of image restoration is to obtain a very close estimate to the ideal image when given a degraded image, the noise, the blurring function and other quality attributes of the original image. The poor quality of the original image is usually caused by blur due to out of focus of the camera, and also the motion blur that can be a combination of geometric transformations (rotation, scaling and translation) [35]. Image enhancement refers to processing techniques that attempts to improve a given image or highlight certain features of interest.

2.2 Related Work

2.2.0 Overview

The section provides a description of FPGA-based implementations and some of the related technologies that are implemented into this thesis. These include discussions of FPGA based frameworks, image enhancement and restoration, edged detection, Xilinx system generator, image subtraction and change detection.

2.2.1 Edge Detection

The research works that have been published in the recent years that associate image processing with re-configurable hardware devices and high throughputs were designed for computer vision applications including image edge detection operation. Various image approaches are reviewed in this sub-section. Edge detection extracts the points in images which correspond to sudden pixel intensity variations or discontinuities [36].

Boo et al [35], designed and implemented the Sobel operator using the ASIC. The parallelism and pipelining features of the ASIC were exploited through systolic array implementation thus allowing

an efficient exploitation of the design. However, this edge detection implementation is limited to a single design unlike an FPGA-based design, which is re-programmable application-specific basis.

Their overall performance was satisfactory and could be applicable in a change detection framework if it was not for the high cost and the limitation to the user's customization. The ASIC devices cannot be reconfigured after manufacturing to suit the user's specific needs or a specific functionality.

Smach et al [37] proposed an approach that uses Simulink/Xilinx system generator design environment and Xilinx Spartan 3A DSP 3400 XC3SD3400A-4FGG676C (FPGA) to implement Sobel edge detection. A hardware implementation of the algorithm targeting the Xilinx Spartan 3A FPGA system was done using model-based block design approach. The image (Lena 512*512) was used as the input image to the Simulink pre-processing unit, and then it was fed to the Xilinx black box via gateway-in. The system generator black box contains a VHDL code for Sobel operator that gets executed and performs edge detection, thus extracting edges in the image.

Basheer et al [38] used Xilinx FPGA to implement Sobel edge detection. The implementation was performed using VHDL on Spartan3E (XC3S1600E) and Xilinx Integrated Software Environment (ISE). The Sobel operator was used to detect edges on an 8-bit gray scale image by applying the horizontal and vertical convolution masks. The MATLAB/Simulink environment was used to generate the bit file to target the Spartan3E FPGA. This approach did not use any image enhancement and restoration techniques. The results obtained in the implementation could have yielded a more superior quality if the input image had gone through denoising and deblurring prior to edge detection. Similarly Yinglin Qin et al [39] implemented a Sobel edge detection using TERAASIC FPGA DE 1 -SoC development board and OpenCL programming which allows the use of C-based programming language for developing code for FPGA. Intel FPGA SDK for OpenCL is adopted in this implementation. The Sobel filters are convolved with the whole input image to compute both the horizontal and vertical edges while the Intel FPGA software development kit (SDK) compiles the kernel file.

Another study was done by Anusha et al [40] who also implemented edge detection using Sobel operator using an FPGA based architecture, an approach that is better compared to software implementation due to its higher processing speed. Here the design is implemented by applying a 3 by 3 convolution kernels to process a 640x480 gray scale image. In addition to the common FPGA based Sobel edge detection, the binary segmentation process was applied.

In a different approach to Sobel edge detection Swain et al [41] proposed an optimized Sobel edge detection. The approach used a histogram stretching to enhance the edges of the image. This edge detection implementation was employed the SPARTAN 3A DSP 3400A development platform FPGA kit. This edge detection design was model based and used hardware co-simulation with the Xilinx system generator. This approach is similar to the proposed edge detection in this thesis though it did not employ all the steps proposed here. The image enhancement and restoration processes performed prior to the Sobel edge detection were the salt and paper noise, and blur removal using the Gaussian filter. Contrast enhancement was used for image smoothening and image sharpening.

Similarly Dixit et al [42], proposed model based Sobel Edge Detection Implementation using Spartan 3 FPGA and Xilinx System Generator. The Sobel edge detection was built using the Xilinx blocks in MATLAB/Simulink environment where VHDL code is generated by the XSG. Image filtering for image enhancement and restoration, which is a pre-processing step for image noise and blur removal, preceded the edge detection implementation. The approach used by Dixit et al, resembles the procedure used in the framework proposed in this thesis, however the authors did not explicitly give a description of the specific image filtering algorithms that were used prior to the Sobel edge detection implementation.

In a different approach Patial et al [43], proposed a Sobel edge detection using parallel architecture based on FPGA. The design was implemented on a binary image in MATLAB generated Verilog code using the Xilinx ISE and Modelsim programs for the synthesizing and simulating the design. The approach used parallel implementation of the horizontal and vertical edges to detect the overall edges of the image for an increased computational speed. A combined pipelining and parallel implementation is recommended for a performance improvement of this approach.

When considering the other type of edge detection; the Canny method, the author Mahalle et al [44] proposed an efficient design for Canny edge detection design using the Xilinx system generator. The edge detection design was implemented using a hardware co-simulation compilation within the MATLAB and Simulink/Xilinx design environment. This approach used the Simulink and Xilinx block sets for model-based design of the edge detection. The design was compiled using a hardware co-simulation compilation driven by the system generator token. The generated bit file from the hardware co-simulation compilation got transferred to the FPGA system (Spartan-3E kit) using the JTAG cable. This approach was efficient in its implementation;

however, the results can be improved if a systematic method of image enhancement and restoration was employed. Gaussian filtering was performed for the removal of image noise prior to Canny edge detection implementation but that was not adequate for a total restoration of degraded images without additional filtering.

Vassiliadis et al [45], proposed a parallel design for real-time Canny edge detection implementation. In this design, the Canny edge detector implementation was performed using parallel architecture on a Spartan 6 FPGA where 4-pixel parallel computations achieved high throughput. The image enhancement and restoration process that was implemented prior was the Gaussian smoothing, which performed image noise and blur removal. Although this Gaussian filtering performed image noise removal, the authors did not use the systematic procedure for denoising, deblurring and image sharpening as outlined in the framework proposed in this thesis.

Martins et al [46], implemented a high-level design for data-flow image processing systems, which targeted a FPGA based platform. The design is a model-based implementation that is built on the MATLAB/Simulink modeling and simulation environment, and the synthesis phase is employed on hardware using Verilog code. The edge detection process implemented in this approach as described by the authors, was successful. However, the image enhancement and restoration operations that are necessary in enhancing the quality of the image through the image noise and image blur removal were not performed. The exclusion of the image restoration and enhancement operations leads to less accurate results since the image blur and other impurities such as image noise affect the image quality. In such a case where the image edges are the features of interest, they will lack contrast and sharpness.

In a somewhat different approach the authors Nguyen et al [47] used a hardware/software co-simulation based system to implement an image edge detection. The Prewitt edge detection operator was performed on the Lena image by using the Xilinx system generator targeting the Xilinx Spartan 6LX FPGA. The design simulation and output waveforms were generated in the Xilinx ISE. The approach used a model-based design in the MATLAB/Simulink development environment. The hardware/software co-simulation utilized by the authors was useful in reducing the validation and verification time, which is an advantage over the FPGA implementation that used VHDL or Verilog code only. The FPGA-based Prewitt edge detection implementation using the Xilinx system generator improved the efficiency and performance of the edge detection.

However, the absence of image filtering had compromised the quality of the output image due to lack of systematic image enhancement and restoration implementation in this approach.

In [48], Tian et al proposed a fast FPGA-based image processing technique that uses two separate parallel operations to increase the speed of median filtering and image edge detection. The parallel architecture is made up of two separate levels of parallelism: the image and the operation level parallels that are used to perform image-processing operations that are highly parallel. The double-parallel architecture improved the performance of the median filter. It also improved the performance of the Prewitt edge detection operation in edge extraction. The authors implemented image smoothing using a median filter and this operation is not adequate to filter both image noise and blur. The filter performs very well only in noise removal, but image deblurring is better achieved by deconvolution algorithms. Image sharpening operations which extract more image details was also excluded. The omission the critical image filtering operations prior to edge detection implementation has was a limitation to the performance of the double-parallel approach.

Tripathi et al [49], proposed an edge detection algorithm using FPGA to extract features of aerial digital images. However, the hardware implementation for feature extraction was based both on the Sobel and Prewitt operators. The model-based design was implemented and verified using Xilinx system generator integrated with the MATLAB/Simulink design environment. In the proposed approach, a color aerial image was converted to grayscale and pushed into the design for feature extraction. The hardware/software co-simulation implementation using the Xilinx system generator increased the performance of the edge detection operation. However, an image enhancement and restoration implementation was not performed for image noise and blur removal. The lack of these image-filtering operations compromised the quality of the output image. A systematic approach for filtering is proposed for a better edge detection output in this thesis.

Selvathi et al [50], proposed the beamlet transform, a robust and efficient transform technique for the edge detection algorithm. The beamlet transform for edge detection was performed on a model-based design that was built in MATLAB/Simulink design environment using Xilinx and Simulink block sets. The implementation was done using the Xilinx system generator, targeting the Virtex5 starter kit (FPGA). The image pre-processing operations were performed, however no carefully ordered and systematic procedure was outlined for these operations. In this approach the authors did not pay attention to the order of the image pre-processing operations. This

limitation may adversely affect the output of edge detection implementation. This is caused by the fact that, when some algorithms are performed in a particular order, the output is adversely affected.

In [51], using a somewhat similar method as in [50], authors Shanshan et al implemented a vehicle image edge detection algorithm using hardware implementation targeting an FPGA. The model-based design was created in a MATLAB/Simulink design environment using the Simulink and Xilinx block sets which was then mapped to the Spartan-3E development board. The approach utilized the Xilinx Integrated Software Environment (ISE), Xilinx system generator for simulation and synthesis. The authors did employ image enhancement and restoration operations that are necessary prior to the edge detection implementation. Exclusion of these operations imposes a limitation in the performance of the edge detection algorithm hence a compromised output quality. The proposed framework in this thesis suggest an created and systematic image enhancement and restoration procedure that enhance and restores a digital images that are in some manner inherently plagued with image noise and blur from acquisition procedure.

Just like the research work in [51], Kavitar et al in [52], demonstrated an FPGA based hardware/software co-simulation implementation of image feature extraction using the Xilinx System Generator for system modeling and generating a bitstream file to be loaded into FPGA. The edge detection was modeled in MATLAB/Simulink design environment using Simulink and Xilinx block sets. The Sobel and Prewitt edge detection algorithms were implemented on Spartan-3E FPGA development board using the Lena image. The authors Kavitar et al did implement image enhancement and restoration processes prior to the Sobel and Prewitt edge detection. This omission of image filtering for image noise and blur removal poses a limit on the effectiveness of the edge detection operations hence an output image with compromised quality. This thesis proposes a sequential and systematic procedure for image enhancement that can bring a drastic improvement to the quality of the edge detection operation.

2.2.2 FPGA-based Frameworks

Salcic and Sivaswamy [13], Benkrid, Crookes and Benkrid [12], and Lingg et al [53] created frameworks that use FPGA hardware for implementing image processing. The change detection framework designed by Bruzzone and Bovolo [54], models the occurrence of distinctive radiometric changes on the basis of the properties of multi-temporal images. This framework also

defines guidelines for the development of a new generation of change-detection methods that can properly analyze multi-temporal, very high resolution (VHR) images. This framework involves pre-processing techniques that may not be used on images that are non-remote sensed. The process of radiometric correction is not required for surface acquired images. The framework limits its model to very high-resolution images, hence cannot be applied to low-resolution images and those that are not sensitive to radiometric changes.

Salcic and Sivaswamy [13], developed a framework to advance the computational speed of image contrast enhancement process using low-cost FPGA, focusing mainly on X-ray images. Their image enhancement method considered and used filtering via a high boost filter (HBF) and a histogram modification using histogram equalization (HE). This framework which is referred to as image enhancement co-processor (IMECO), enabled efficient image enhancement implementation using hardware/software co-design to achieve high performance using low-cost solutions. IMECO only deals with improving image enhancement using the two mentioned techniques, and some enhancement algorithms were not explored. It only targets medical imaging, especially X-ray images. The framework is not generic and could not be used for other image processing applications, which do not necessarily have to use the histogram equalization algorithm. However, the high boost filter (HBF) algorithm is adopted and incorporated in this research for the generic image enhancement processes that is applicable to other applications besides medical imaging.

Benkrid, Crookes, and Benkrid [12], present a framework for FPGA-based image processing. This framework uses the Hardware Skeleton Library that includes high-level instructions optimized exclusively for the XC4000 FPGAs. The authors introduced the concept of hardware skeletons, which is a set of specific instructions with different parameters. For instance, distinctive skeletons for the same operation can be assigned to different arithmetic representations. This makes various implementation alternatives available for users. The hardware skeletons make it possible for a user to provide parameters such as values and functions. This framework targets a single-chip FPGA system and it is not generic in nature hence its use on other FPGA systems is greatly limited. The framework is not flexible, and its hardware skeleton library does not contain configurations that allow dynamic reconfiguration. This image-processing framework cannot handle the many image processing techniques with differing requirements for various algorithms.

The change detection approach proposed by Lingg et al [55], is based on the use of multiple image pairs of the area of interest or same scene in a sequence instead of a single pair of images.

The main purpose of this image sequence approach is to improve the accuracy of the change detection by using more image pairs. Change detection performance is improved with additional measurements as more image pairs are analyzed and data is updated continuously.

The limitation of the framework is the use of a sequence of image pairs which is not very applicable in many applications such as product inspection, surveillance, X-ray and other medical diagnosis processes, a single pair is adequate to implement a change detection process on it. The research carried out with regard to the proposed framework was focused on a single pair of images to detect change. The nuisance or insignificant changes such as sensor noise, camera motion, illumination were taken into account because the research work for this FPGA-based change detection largely covers the usage of image pre-processing algorithms as a pre-requisite stage prior to performing change detection processes. The literature in the work by Abdul Manan [56], describes a generic framework for image processing based on FPGA. This literature discusses FPGA hardware implementation but only focuses on using MATLAB and FPGA to perform morphological image processing algorithms that transform the geometric structure of an image. This framework also lacks the systematic image enhancement steps that form a vital pre-processing stage in image processing, particularly change detection.

Table 2. 1 Literature review for change detection frameworks

	Author, Year	Title	Approach	Pros	Cons	Gap/Limitations
1	Lorenzo Bruzzone, Francesca Bovolo, 2013	A Novel Framework for the Design of Change Detection Systems for Very High Resolution Remote Sensing Images	Framework models the presence of different radiometric changes on the basis of the properties of multi-temporal images	Extracts the semantic meaning of radiometric changes, identifies changes of interest with strategies designed on the basis of the specific application.	This framework involves preprocessing techniques that may not be used on images that are non-remote sensed. The process of radiometric correction is not required for surface acquired images	Framework tailored for multitemporal images. The framework is not generic so cannot be used on other types of images.
2	Z.Salcic, J.Sivaswamy, 2008	IMECO: A Reconfigurable Fpga-Based Image Enhancement Co-Processor Framework	Developed a framework to improve the computational speed of image contrast enhancement using low-cost FPGA based hardware primarily targeted to X-ray images	The high boost filter (HBF) algorithm is adopted and incorporated in this research for the generic image enhancement processes that is applicable to other applications besides medical imaging.	Framework is not generic and could not be used for other image processing applications, which do not necessarily have to use the histogram equalization algorithm.	Other contrast enhancement techniques for X-ray images can be explored.
3	K Benkrid, D Crookes, A Benkrid, 2002	Towards a General Framework for FPGA Based Image Processing using Hardware Skeletons	Hardware Skeleton Library which contains a set of high-level descriptions of task-specific architectures specifically optimized for Xilinx XC4000 FPGAs.	Skeletons are parameterizable, and different skeletons for the same operation can be provided, giving the user a range of implementation choices.	This library contains optimized implementations tailored for Xilinx XC4000 FPGAs	This framework it is optimized for a single FPGA embedded system, which greatly limits its use on other FPGA system

Table 2. 1 Literature review for change detection frameworks, continued ...

	Author, Year	Title	Approach	Pros	Cons	Gap/Limitations
4	Andrew J. Lingg, Edmund Zelnio, Fred Garber, Brian D. Riglin, 2014	A Sequential Framework for Image Change Detection	Change detection is performed on multiple image pairs of the area of interest or same scene in a sequence instead of a single pair of images. This approach also catches changes that occurred between the passes.	Change detection performance is improved with additional measurements as more image pairs are analyzed and data is updated continuously	The framework cannot be implemented in applications where a sequence of images or video frames is not required such as in medical images where a single pair of images is adequate.	The framework is not generic in that many applications need a single image pair for change detection. The main purpose of the sequence is to improve the accuracy of the change detection by using more image pairs.
5	Javier López-Fandiño, Dora B. Heras, Francisco Argüello- Mauro Dalla Mura, 2017	GPU Framework for Change Detection in Multitemporal Hyperspectral Images	Graphic Processing Unit (GPU) based change detection framework tailored for change detection in remote sensed images using object-based algorithms	The integration of the CUDA GPU implementation in the framework provides parallel computation for increased processing speed. The framework uses a multi-GPU system that can be implemented for various change detection techniques	The framework is not generic but tailored for hyperspectral images whose time frame is normally in several months. This framework is not suitable for other image types that may be produced in short time frames like the video frames.	The segmentation process used is suitable for hyperspectral images and cannot be used for generic image processing.
6	Abdourrahmane M. Atto , Grégoire Mercier , Dominique Pastor, 2009	General Framework On Change Detection In A Sparse Domain	A turbo-median based framework for change detection in remote sensed images.	The use of the turbo-median filtering technique greatly reduces noise especially speckle noise in the SAR data.	The framework works for the remote sensed images and was not implemented on other types of digital images. The process for removing blur and image registration for SAR images were not implemented in this framework.	The approach is tailored for a specific type of data and there is no systematic pre-processing procedure that can be applied to other types of data.

Table 2.1 Literature review for change detection frameworks, continued ...

	Author, Year	Title	Approach	Pros	Cons	Gap/Limitations
7	Nicolas Seiller, Nitin Singhal, and In Kyu Park, 2013	Object Oriented Framework For Real-Time Image Processing On Gpu	A GPGPU-based image processing framework that uses object oriented programming. This approach uses both the graphic processing unit and central processing unit systems programming resources	OpenGL Shading Language (GLSL) and NVIDIA Compute Unified Device Architecture (CUDA) are incorporated in the GPU-based framework for image processing algorithms that demands high processing power	The framework uses object-oriented implementation on GPU and performance is affected by the overhead function calls.	Sequential programming procedure limits the GPU performance on image processing algorithms. The framework lacks hardware implementation computational intensive processes
8	Abdul Manan, 2006	Implementation of Image Processing Algorithm on FPGA	A generic framework for image processing based on FPGA	A systematic image processing framework using MATLAB and FPGA	A morphological image processing might not be necessary in other image processing applications since algorithm changes the geometric makeup of an input image	No systematic pre-processing steps are described for image processing.
9	Philippe Guermeur, 1998	A New FPGA Architecture For Image Processing : Cyclop	A multi-FPGA board architecture designed for real-time hardware implementation of image processing.	FPGA system comprising a general-purpose computer and multiple FPGA boards, all integrated into a single system for improved image processing speed.	Need to partition image processing algorithms for multiple FPGA boards. If high performance is required, a pipeline processing has to be implemented by the user.	The limitation is that a synthesis process for a hardware implementation was not included.

Table 2. 1 Literature review for change detection frameworks, continued ...

	Author, Year	Title	Approach	Pros	Cons	Gap/Limitations
10	Jayalakshmi S Nair, Bibin Binu Simon, 2014	An FPGA Based Image Denoising Architecture with Histogram Equalization for Enhancement	An efficient denoising technique and hardware implementation for the removal of noise	Denoising affects only the noisy pixels of the image, and all other pixels are left untouched. This technique preserves the edges of the image and improves the quality of the images before change detection algorithms can be applied.	Programming logic devices involves complex power requirement configuration and the hardware description languages (VHDL and Verilog) have a steep learning curve. These factors pose as disadvantages to this approach	The only VLSI architecture implemented is the low complexity; low complexity could not be implemented.
11	M Rao, P.R kumar, A Mallikarjuna, 2016	Image Processing using FPGAs: A framework	A MATLAB based framework implementing image processing algorithms	This framework provides a general approach that gives a systematic procedure for processing images including pre-processing.	The framework is not scalable and is less generic in that it uses a Xilinx generator, proprietary library that is not available in other FPGA systems.	The hardware implementation is tailored for Xilinx system and giving a limitation for other systems. Processes such as converting data to 1D could be done in a generic architecture.
12	J. Fegan, S. A. Coleman, D. Kerr. B. W. Scotney 2-017	An Implementation Framework for Fast Image Processing	A fast image processing approach based on the Spiral Addressing Scheme in place of the common 2D Cartesian address	When a one-dimensional structure is represented as a 2D structure with the use of spiral addressing scheme, convolution is performed in the vertical direction rather than in the horizontal direction. This provides fast algorithm computation.	This is a new technique that showed high performance and effectiveness on image processing, but it has not thoroughly test with many algorithms that are applied in various image-processing applications.	Some algorithms are limited and cannot be fully implemented where in some cases Linear approach can perform a convolution of operators of size 3x3 only

Table 2. 1 Literature review for change detection frameworks, continued ...

	Author ,Year	Title	Approach	Pros	Cons	Gap/Limitations
13	Mohammad I. Ali, Khaldoun M. Mhaidat , and Inad A. Aljarrah, 2013	Implementing Image Processing Algorithms in FPGA Hardware	Implementation image processing Algorithms and filtering techniques using FPGA	Generic preprocessing techniques that are applicable to both video and images are discussed. The implementation process utilizes the generic Windowing Operations for calculating the pixel out.	The preprocessing techniques are not all covered to cater for images in general. Image processing techniques like deburring, sharpening, color and contrast adjustments were not included in the pre-processing stages	The preprocessing procedure is incomplete and needs to include more enhancement techniques
14	Mohammad R Mohmoodi, Sayed M. Sayedi , 2016	A Low Cost High Speed FPGA-Based Image Processing Framework	Developing a MATLAB-FPGA communication link that provides a point-to-point FPGA-PC communication which can be directly utilized in implementation of hardware-software co-designs	The FPGA-PC interface is proposed that can be used in developing and prototyping many FPGA based DSP algorithms and it provides a remarkable solution to FPGA-MATLAB communication for variety of applications.	Framework is not generic and does not provide a systematic image processing procedure for image change detection. The proposed framework was then used for implementing a skin detection algorithm and motion detection	The framework is not tailored for image processing methods such as image enhancement, image restoration, edge detection that are necessary for image change detection process.

2.2.3 Chapter Summary

Some research studies have been done on field programmable gate arrays based change detection frameworks and various image-processing methods have been implemented. Although the literature is extensive and covers various techniques, and a wide range of frameworks, this literature review has focused on generic techniques that use field programmable gate arrays for implementing image processing. These include image enhancement processes, image restoration and change detection techniques. Although most literature studies present change detection frameworks for vendor-specific and frameworks tailored for specific applications, this research has primarily focused on generic FPGA-based change detection framework. Several researchers have implemented change detection frameworks using FPGA with the aim of realizing results in the fastest time possible, at low power and with reduced cost in image processing operations. The frameworks published in literature do not present any adequate generic and systematic procedure for implementing image change detection using FPGA-base framework.

Chapter 3: Framework Modeling and Implementation

3.0 Overview

This chapter discusses the modeling of the generic framework for FPGA-based change detection proposed in this research. In this chapter, the methods developed for performing image processing are presented and detailed descriptions of the major components of the image-processing framework are given. The proposed framework is built from model-based block sets in the MATLAB/Simulink design environment targeting the FPGA system. The implementation of the modules of the change detection framework targeting the Xilinx FPGA system was done using model based block design approach. The MATLAB M-codes for algorithms and VHDL programming were used for implementation of the framework.

3.1 Proposed FPGA-based Change Detection Framework

The design flow of the proposed FPGA-based framework is presented in Figure 3.1. The generic framework consists of six main phases, which are image restoration and enhancement, image registration, edge detection, image change detection, contrast stretching and post processing. The image registration stage is optional and can be omitted except for the input images require alignment. The image processes for the change detection framework require a lot of computational power for the several image processing algorithms involved, especially the edge detection and image change detection. The level of computational power increases tremendously when high-resolution images and raw images are involved. A reconfigurable computing technology, which is field programmable gate array (FPGA) based has become a viable platform for the implementation of algorithms suited for image processing applications. The reconfigurable property of the FPGA is its ability to alter the functionality of the circuit to suit any specific application. For this reason, the FPGA system is incorporated in this framework to cater for these computationally intensive operations by exploiting its parallelism and high digital processing performance properties.

The key phases of the proposed generic change detection framework are shown in Figure 3. 1. The framework takes two inputs, the first-date image (1st date image) and the second-date image (2nd date image). The first phase is the image restoration and enhancement, this is where the image quality is enhanced and restored to their original state. When images are acquired, there are impurities such as noise and blur that get included into the acquired images due to various environmental conditions such the quality of the camera lenses, the position of the lenses, and the shaking of the equipment in use. The aim of this phase is to remove the noise and other impurities that might affect the success of other operations in the framework. The second phase is the image registration which is very important in some applications where some regions or objects in the input images need to be aligned. This phase is optional when the input images are in good alignment.

The third phase in the framework is the edge detection process where the lines, corners and edges in input images are extracted. Edge detection increases the effectiveness of image change detection, which is the next stage in the framework. The change detection is the core of the framework, and its main purpose is to spot the differences between two input images. The change detection phase has a single output, which is the difference image. Following the change detection is the contrast-stretching phase whose main function is to extract the fine details of the output image. This process prepares the output to suite visual interpretation. The last phase of the framework is the image post processing, which is mainly, image display.

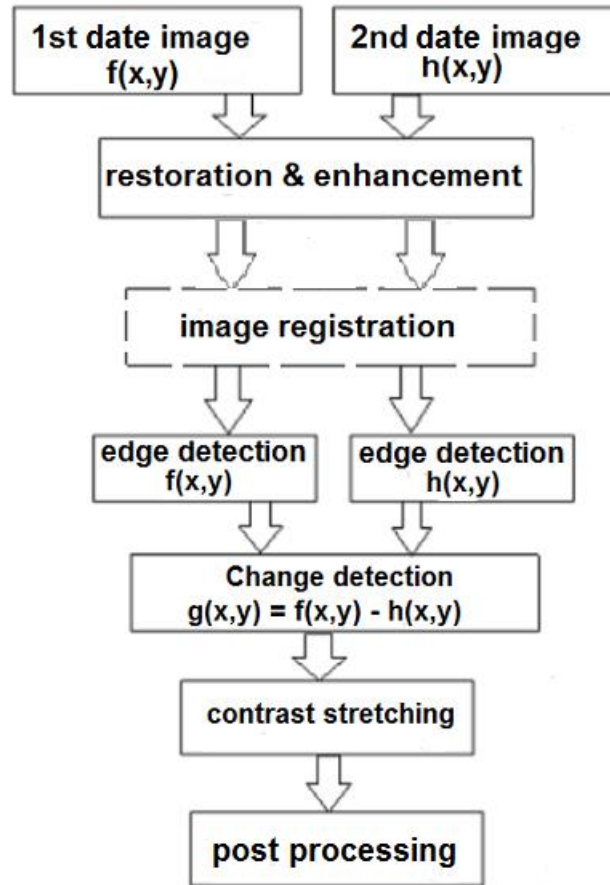


Figure 3. 1 The proposed FPGA-based change detection framework

3.2 Components of the Proposed Framework

The proposed framework comprises six components or phases of image processing operations and these are image acquisition, image enhancement and restoration phase, image edge detection, image subtraction and lastly the contrast stretching operation. In section addresses and answers research questions (iii), (iv) and (v). A generic block diagram showing all key image processing design blocks connected together in a sequential order is depicted in Figure 3. 1. The modules that make up the image change detection framework have a specific order of arrangement within the framework. The first module or phase of the framework is the image

enhancement and restoration, which is divided into three phases; the image denoising, image deblurring and image sharpening. These operations are implemented in this respective order starting with denoising, followed by deblurring and then image sharpening. Denoising or noise removal is performed before other operations since image noise prevents progress the most in image processing and can affect the effectiveness of other algorithms when it is removal last. This process is then followed by blur removal which restores the image to a close estimate to the original and it works better when the image has less noise. Blur removal improves the quality of the image and when image sharpening is performed on the image, finer details are extracted, making the image sharpened. Adherence to this order of image processing operations is pivotal in this framework because the sequential order of operation of the image processing modules crucial for retaining and preserving the image quality.

The module that follows immediately after enhancement and restoration is the edge detection, which then connects to the image subtraction phase, and the last phase is the image contrast stretching as illustrated in . All these key stages of image processing operations are linked and connected together to form the proposed change detection framework.

The two inputs to the framework are 1st date image and 2nd date image. Each image is passed through the restoration and enhancement steps of the framework. The first step is the denoising, where an appropriate algorithm is performs noise removal. The step that follows is the deblurring stage where image blur is removed. The images then go through the sharpening process where the fine details of the images are revealed. The stage that follows is edge detection where the image lines, corners and edges are extracted. This is the last process in the image restoration and enhancement phase.

The next stage is the binary conversion process that highlights the key details in the image to aid in further image processing operations. The image data is now in binary format and passed to the change detection process. This image process takes the two inputs, the image data for the 1st date image and 2nd date image and a change detection technique is applied to detect the differences between the image data. The out of this stage is a binary format image data which is converted to two-dimensional format and then passed to the image contrast stretching process which improves the image for easy visual interpretation.

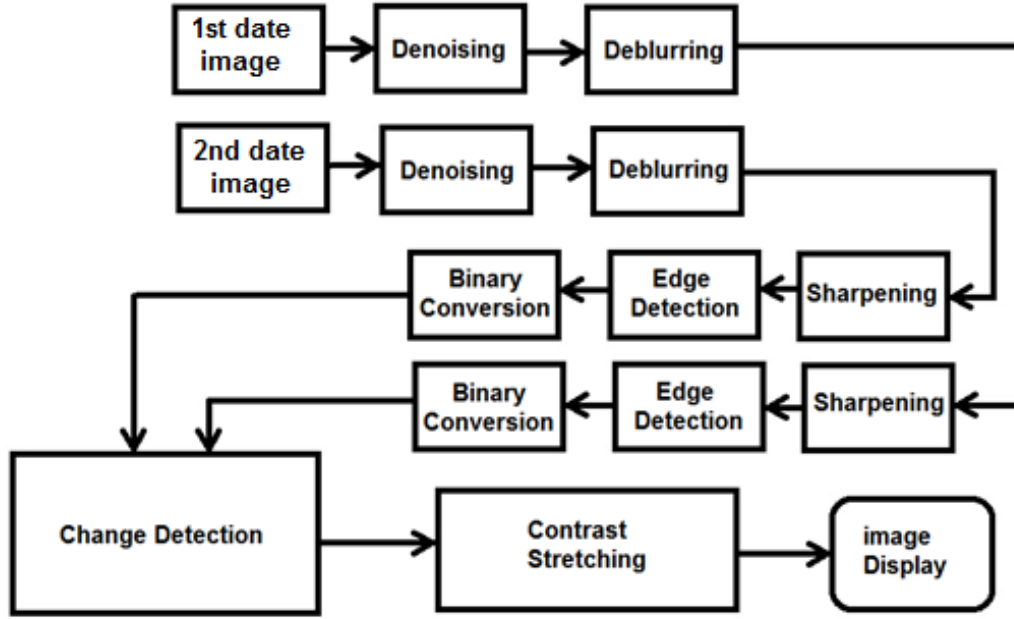


Figure 3. 2 Key stages of the proposed generic change detection framework

3.2.1 Image Acquisition

Image acquisition is one of the fundamental steps in digital image processing. It is a process of creating digital images, usually from a hardware based source [57].

The images or video acquired from a digital camera are already in digital form, and video files can be processed just like still images by decoding their frames and processing them as still images. Images acquired from other pieces of equipment such as scanners, must be digitalized both spatially and in amplitude for them to be suitable for analysis and computer processing. These images $f(x, y)$ and $h(x, y)$ are the inputs to the image pre-processing stage in the FPGA change detection system as shown in Figure 3. 2

3.3 Image Enhancement and Restoration

The main purpose of image enhancement and restoration is to improve the quality of images in ways that will increase the chances for success of other image processing operations that follow. Image enhancement and restoration processes are also popularly referred to as image pre-processing and they take place over several steps. In order to achieve a high-level efficiency in a change detection process, it is necessary to perform image enhancement and restoration in the input image data. There are some overlap properties that exists between image enhancement and image restoration. Image enhancement is mostly a subjective process and image restoration is an objective process and has a quantifiable goal. The primary goal of image restoration is to recover an image that has been degraded and it involves the modeling of degradation and applying the inverse process in order to recover the image. There are numerous reasons for performing image enhancement and restoration on images stipulated below as:

- (i) Highlight regions of interest in the images
- (ii) Noise removal
- (iii) Make images more appealing for visually purposes
- (iv) Extract hidden features in images
- (v) Filter discarding less important features in the image

Image enhancement and restoration processes can be described either as spatial domain techniques or frequency domain techniques [58]. In the frequency domain, the image enhancement processing techniques involve the manipulation of the Fourier transform of an image; there is no direct processing of intensity of each image pixel. The spatial domain techniques are performed on image plane itself and deal with direct manipulation of the image pixels. In this work the spatial domain processes are denoted by the expression

$$g(x, y) = T[f(x, y)] \quad 3.1$$

where $f(x, y)$ is the input image, $g(x, y)$ is the output image, and T is an operator on f , defined over some neighborhood of (x, y) [59].

There are three classifications for the spatial domain; the point processing, spatial filtering and image coloring. Point Processing refers to all processing that can be done based on a single image pixel. This image enhancement technique applies a transformation to individual pixels to produce new images [60]. Some of the techniques that use point processing are histogram modeling, contrast enhancement and image averaging. Image coloring refers to pseudo coloring or false coloring, a technique used to assign color a gray scale image or to modify an existing color image giving it a new color. This image restoration technique improves the visual quality of the image by highlighting specific features in the image [60]. The third classification is the spatial filtering, which is defined as a technique that performs discrete implementation of convolution filtering by placing a window or a kernel on a pixel and process only the image pixels that fall within the neighborhood. This process is repeated over the whole image to compute new intensity values for all the pixels, and the main purpose of the filtering algorithms is to enhance the image.

A filter kernel is a set of coefficients $w(s, t)$ where (s, t) runs over small neighborhood N of the origin:

$$N = \{(s, t) : -a \leq s \leq a, -b \leq t \leq b\} \quad 3.2$$

$$G(x, y) = \sum_{-a}^a \sum_{-b}^b w(s, t) f(x + s, y + t) \quad 3.3$$

$f(x, y)$ is the input image. The output image $G(x, y)$ is the product of summation of the pixels values and filter coefficients.

Spatial filters can be classified as linear and non-linear filters [61], [62]. These two types are utilized in creating the change detection framework depending on the type of noise; therefore the image enhancement and restoration operation stages of the framework are performed using both linear and non-linear spatial filtering techniques. The framework is tailored for a change detection procedure that involves small size, large size and high-resolution images. Since there are many image enhancement and restoration processes that perform similar enhancement functions, a few have been carefully selected to suit a wide range of applications for the FPGA-based change detection framework.

In this image change detection framework, image enhancement include; denoising, deblurring, and sharpening as shown in Figure 3. 3. Image enhancement and restoration, and the change detection algorithms are performed by the FPGA system in the order depicted in Figure 3. 3. The first three Image enhancement and restoration steps described in this framework perform image noise removal, blur removal and sharpening. The image registration process is optional, it can be used when need be, to filters out common “unimportant” intensity changes before a change detection process could take place.

The order of operations and the techniques applied in image enhancement are very crucial and can have a profound impact on the output. Therefore, the arrangement of the image enhancement and restoration steps in the flow diagram Figure 3. 3, is the exact order in which the image enhancement and restoration should be executed in the framework. The same order also follows in the detailed description of the framework in this document as outlined in this chapter. The proposed order of image enhancement operations for image enhancement, presented in this chapter focuses on an order of operations for images that require multiple image enhancement techniques. However, the effectiveness of these techniques is determined by different aspects, for instance the spatial resolution of the image, and not all images require all these techniques to be performed. In some cases, it would not be required to apply specific enhancement algorithms depending on the type and quality of the image file. In such an event, an image enhancement technique in the operations order may be skipped and the next technique would be carried out.

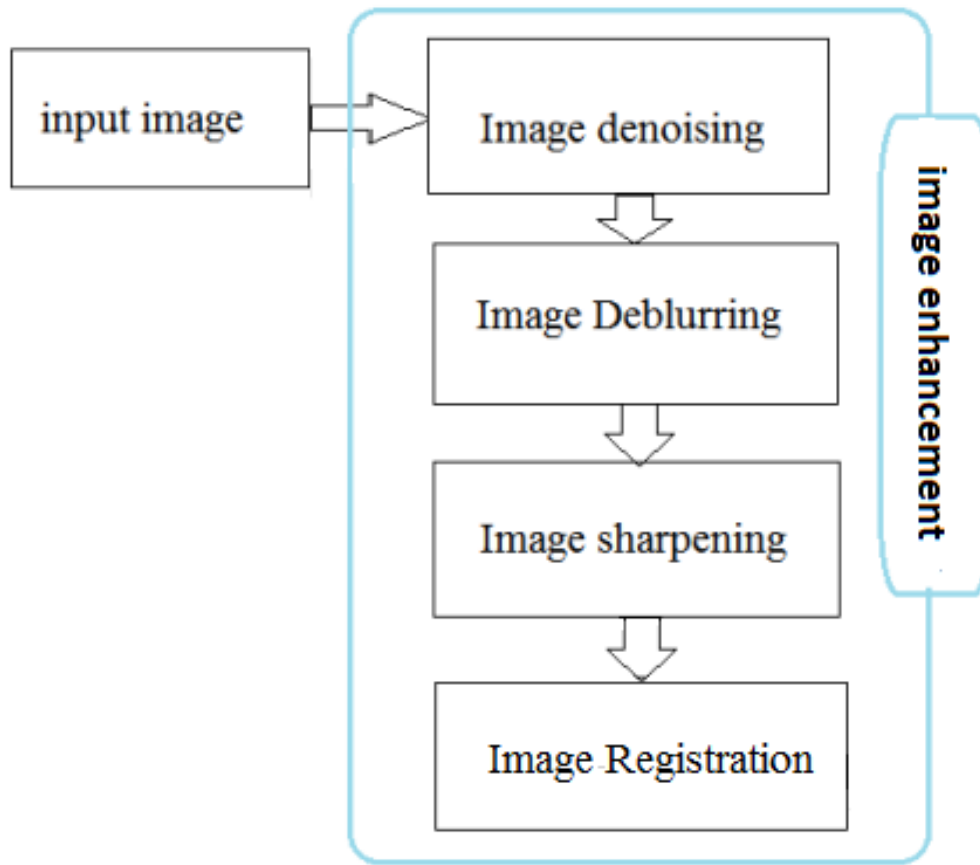


Figure 3. 3 Key steps of image enhancement and restoration

3.3.1 Digital Image Denoising

Denoising is a process used for noise removal in digital images. Noise is any unwanted signal present in the original signal [63]. Noise brings a lot of problems in image preprocessing because it interferes with background scenes of the image and creates image-degrading effects such as blurred objects, artifacts and unrealistic edges.

The denoising phase describes impulse noise and its removal from digital images. It exists in several forms and it is caused by different environmental conditions and image acquisition equipment. Impulse noise formation in most cases occurs during image acquisition period, during transmission or storage. Image noise is an undesired signal in digital image processing due to the fact that it adversely affects the quality of the output images produced by digital image

processing algorithms. This noise poses a serious challenge in digital image restoration, and its removal is achieved by implementing either spatial domain or frequency domain techniques.

Noise has very adverse effects on image compression and image analysis. For this reason, the denoising process is implemented before the other three enhancement operations because when an image is not denoised, the noise in the image interferes with other processes such as image sharpening and deblurring where high frequency signals are amplified [64]. In such a case, the noise will also be amplified together with the high frequency signals, therefore lowering the quality of the image.

The process of denoising can be performed using image-filtering processes. There are two image denoising methods; the spatial filtering and the transform domain filtering [66]. The Spatial filters can be classified into two types; the linear and non-linear filters. There are several linear and non-linear filtering algorithms that have been developed to remove all kinds of noise from corrupted digital images to enhance the image quality. Some of the common image noise types are the Gaussian noise, salt and pepper noise, Poisson noise, and Speckle noise.

The selection of the image filtering methods and algorithms depends on the type of noise that has to be removed from the digital images. For instance, non-linear filters are commonly used to remove impulsive noise, which can be classified as random valued noise or “salt and pepper noise.

3.3.2 Image Deblurring

Deblurring is a process of removing image blur by recovering a sharp image from the corresponding degraded image. Image blur is usually caused by scene or subject movement, shaking of the camera and inaccurate focusing, depending on the depth of field [66]. This pre-processing technique is necessary for enhancing the images so that further image processing algorithms can be more effective without being limited by blur effects. In this framework, a deblurring process (high-pass filtering) will be demonstrated on a blurred image. When an image is distorted, any changes that had occurred in the image cannot be properly detected. When an image is affected by a blur, other image enhancement techniques such as sharpening and colour adjustments cannot be effective; therefore blur removal must be performed before sharpening to avoid amplifying the blur in the image.

Image blur can be classified in three types: average blur, which is usually removed using the average filter. The second type of image blur is the Gaussian blur, which is generated through Gaussian function; it is dense at the centre and lighter at the edges. The third type of blur is the motion blur, caused by movement of the camera and the scene [67]. Image deblurring techniques can be classified into two categories, the blind deconvolution and non-blind deconvolution [68], as depicted in Figure 3. 4.

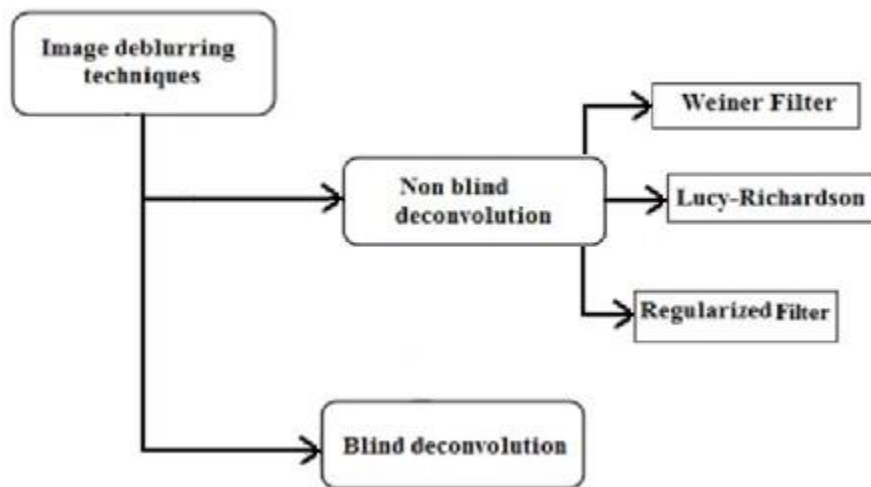


Figure 3. 4 Image deblurring techniques

Blind deconvolution refers to a more practical approach where the blur kernel is unknown. In this case either the sharp image or the kernel or both have to be estimated from the deblurred image [69]. The non-blind deconvolution can be further classified into three categories; a deconvolution that uses the Wiener filter, a deconvolution that uses the Lucy-Richardson method and a deconvolution that uses a regularized filter. In non-blind deconvolution a pre- determined blur kernel value is used, and a sharp image can be induced from both the blurred image and the kernel. In this image enhancement step, the deblurring discussion is limited to the deconvolution that uses the Wiener filtering. A clear and sharp image can be obtained from a blurred image by using the common filtering methods; a modified inverse filter and the Wiener filter. A blurred image is a type of degraded image and caused by additive noise.

Image blur diagram is shown in Figure 3. 5 and it can be modeled as

$$g(x, y) = h(x, y) \times f(x, y) + n(x, y) \quad 3.4$$

where $h(x, y)$ is the blurring or degrading function, the distortion operator commonly known as point spread function (PSF) [70].

$f(x, y)$ is the original non-blurred source image, $g(x, y)$ represents the blurred image and $n(x, y)$ is additive noise introduced during image acquisition.

This method requires an estimated or prior knowledge of the parameters of point spread function and if the PSF is known, then the reverse process can be performed to remove the blur. The sharp image $f'(x, y)$ is derived from $g(x, y)$.

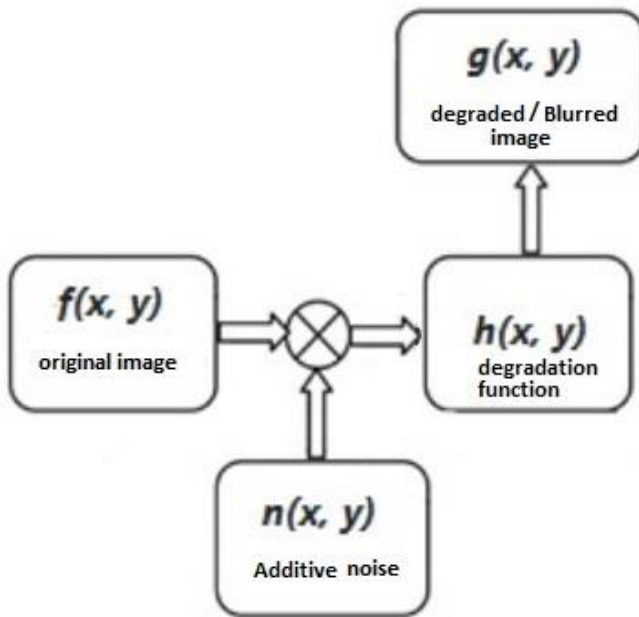


Figure 3. 5 degradation model of image

Expressing equation 3.4 in the frequency domain:

$$G(u, v) = F(u, v)H(u, v) + N(u, v) \quad 3.5$$

If we let $T(u, v)$ be a reconstruction filter,

$$F'(u, v) = G(u, v)T(u, v) \quad 3.6$$

Where filter $T(u, v)$ finds the best approximation of the original non-blurred image.

$$T(u, v) = H^{-1}(u, v) \quad 3.7$$

where $H(u, v)$ is the sensor function

Using the inverse filter for deconvolution produces an image that is distorted due to lot of noise which is caused by a very frequency gain where $H(u, v)$ is small. However, an inverse filter is improved to prevent the image distortion. The high-frequency gain is reduced hence lowering the noise amplification. In the modified filter, Butterworth response is used. In order to avoid distorting the output image, an appropriate Butterworth response was picked [71].

$$B(u, v) = \frac{1}{1 + \left(\frac{u^2 + v^2}{D^2}\right)^n} \quad 3.8$$

$$T(u, v) = \frac{B(u, v)}{H(u, v)} \quad 3.9$$

$$\check{F}(u, v) = (F(u, v)H(u, v) + N(u, v))T(u, v) \quad 3.10$$

$$= F(u, v)B(u, v) + \frac{N(u, v)B(u, v)}{H(u, v)} \quad 3.11$$

3.3.3 Image Sharpening

Image files from cameras are always slightly blurred because every process of image capturing introduces some type of blur. Image sharpening accentuates image features and improves the focus making details appear much clearer. It is normally achieved through the use of high pass filtering [72]. A spatial high-pass filter removes blurring, extracts edges and other discontinuities from images, thus making image details come out clearly. It also passes only lets through the high frequencies while reduces low frequency components such as lines, edges and points without disturbing high frequency information [73]. The high pass filter skips the regions in the image that are not edges and apply the sharpening algorithm on those regions with edges. Some of the sharpening algorithms used in sharpening digital images are the Laplacian filtering unsharp masking and high boost filtering. For illustrating the sharpening process, unsharp sharpening method was used.

Unsharp masking can be described as a method for sharpening images. It uses a slightly blurred version of the original image to create a mask, and then the blurred image is subtracted from the original image itself [74]. The image difference is then added to the original image to obtain a sharpened image, shown in Figure 3. 6.

If the original image is denoted by $f_o(x, y)$, with x coordinate and y coordinates.

And the blurred version of the original image is denoted by $f_b(x, y)$,

Then the image difference between the original and the blurred image, which gives the mask, is calculated using equation 3.12 [75]. The weighted portion of the mask is denoted by Ω

$$f_d(x, y) = f_o(x, y) - f_b(x, y) \quad 3.12$$

$f_s(x, y)$ is the sharpened image

$$f_s(x, y) = f_o(x, y) + \Omega \times f_d(x, y) \quad 3.13$$

Equation 3.13 gives the unsharp masking when the weighted mask equals 1, and when the mask is greater than 1, high boost filtering (unsharp masking) is achieved by adding part of the original image back to the high pass image. The high-boost image looks very close to the original image with details of the edges improved, [81].

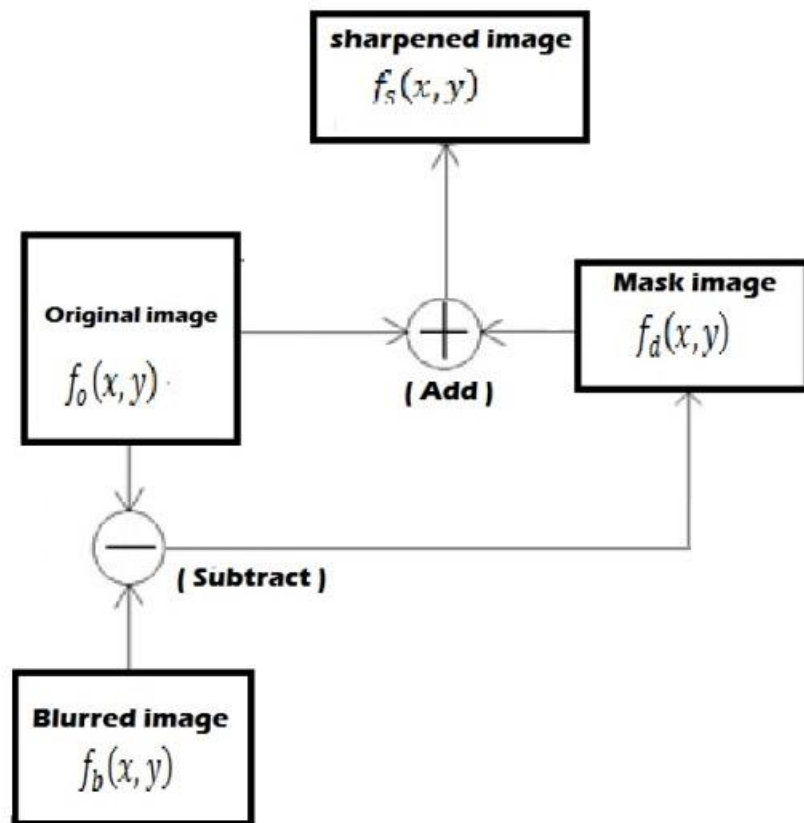


Figure 3. 6 Image sharpening using unsharp masking

3.3.4 Image Registration

Image registration is the process of aligning two or more images of the same object or scene taken at different times, from different sensors, or from different viewpoints into a single integrated image for better observation. The practical applications areas of image registration can be categorized into four classifications; multi-temporal analysis, multi-view analysis, multimodal

analysis and the scene to model analysis [76]. Image registration provides mapping of corresponding points between two images, hence aligning the images geometrically, or expressing the images in the same coordinate system. In this phenomenon, each pixel position corresponds to the same part of the scene in the other image. At times when the points do not coincide, the images are distorted in order for them to match. Image registration is used to resolve problems such as image skew, rotation and scale in overlaying images. Image registration is a significant pre-processing technique, which usually finds its application in environmental monitoring, astronomy and medical imaging [77]. In these applications, input images have to be in the same reference coordinate system for accurate image processing and analysis purposes. In this change detection framework, image registration is optional more especially when the scene or object of interest is stable with little camera movement. If it happens that an unstable or shaking imaging equipment for instance causes image intensity changes, an image registration process must be performed because such changes cannot be regarded as real changes in a change detection process. However, image registration is not an option and is very crucial when this framework is to be applied in multimodal analysis because image alignment is critical in areas such as image mosaicing. The implementation of image registration can be done using four steps namely; transform model estimation, resampling and transformation, feature detection and feature matching.

There are various criteria used in selecting an image registration technique and there are three approaches: feature-based matching, intensity-based matching and the hybrid approach. These criteria largely depend on the kind of features or type of image degradation, radiometric or geometric deformations and application areas [78].

3.4 Edge Detection

Edge detection detects sharp discontinuities or sudden changes in image pixel intensity and highlights boundaries of objects in a gray scale image. The edge detection process produces line features of the image that are then used by advanced image processing algorithms for further processing of the image [79]. Edge detection algorithms can be classified into two categories; the

first-order derivative edge detection algorithm (gradient based operators) and the second-order derivative edge detection algorithms (Laplacian based operators) [79].

Some of the practical edge detection operators are the Robert Detector, Prewitt Detector and Sobel Detector. These are first order derivative operators and they use different kernels. The focus of this chapter is on the Sobel edge detection operator, which is commonly used mainly for its advantages on being less sensitive to noise and having a smaller mask compared to the other two operators. The Sobel operator is an edge detection algorithm that performs a directional smoothing conjugate with a gradient calculation in the orthogonal direction [80]. The Sobel operation performs both edge detection and binary conversion. There are two convolutions used by this Sobel operator, the 0 and 90 degrees convolution kernels and gradient magnitude at each point is calculated by adding the absolute individual kernels (k_x, k_y) [81]

In this thesis, edge detection process comes at the end of the image enhancement and restoration phase of the framework. There are two main purposes of the edge detection process in this framework; edge feature extraction and thresholding, which converts the gray scale input images to binary format. The binary conversion is also a necessary process that prepares the input images for XOR operation, which performs image subtraction in the FPGA system.

The algorithm implemented at this step produces binary versions of images $f(x, y)$ and $h(x, y)$. These binary images appear as a combination of horizontal and vertical line drawings depicting all edges in the images.

The first-order derivative is calculated using a pair of 3×3 convolution masks or filters, one for the x-direction gradient (for detecting horizontal edges) and the other for the y-direction gradient (for detecting vertical edges).

The Sobel convolution kernels for the vertical direction (k_y) and horizontal direction (k_x) are expressed as follows; equation 3.14 and 3.15 respectively.

$$k_y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix} \quad 3.14$$

$$k_x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \quad 3.15$$

Edge detection with a Sobel operator is implemented by filtering a gray scale image with the k_x kernel and the k_y kernel, then the pixels values of each filtered image are squared. The edge magnitude is then estimated by taking the root of the total sum of the squared values [82].

Therefore, the edge magnitude can be estimated as

$$G = \sqrt{(k_x^2 + k_y^2)} \quad 3.16$$

And it can also be calculated as

$$G = |k_x| + |k_y| \quad 3.17$$

Edge gradient direction is estimated as

$$\theta = \text{atan}\left(\frac{k_y}{k_x}\right) \quad 3.18$$

If we let I be the original image, kernels k_x and k_y are used to calculate the corresponding derivative approximations

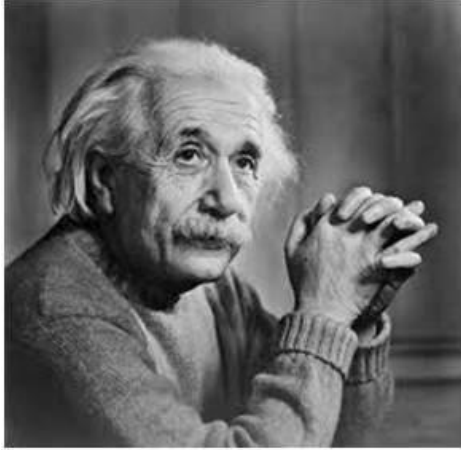


Figure 3. 7 Original source image

3.4.1 Vertical Edge Detection

The Sobel edge operator uses the G_y kernel to convolve with the original image to calculate approximations of the derivative for vertical changes [49].

If the image with the vertical edges Figure 3.8, is denoted by G_y then this image can be expressed as;

$$G_y = k_y \times \mathbf{I} \quad 3.19$$

Where $\mathbf{I}$ is the original image Figure 3. 7, [83] and k_y is vertical kernel.



Figure 3.8 Image with vertical edges

3.4.2 Horizontal Edge Detection

The approximations of the derivative for the horizontal edges are calculated by the convolution of the original image with the G_x kernel.

If the image with the horizontal edges Figure 3.9, is denoted by G_x then this image can be expressed as;

$$G_x = k_x \times \mathbf{I} \quad 3.20$$

Where $\mathbf{I}$ is the original image and k_x is vertical kernel [49].



Figure 3.9 Image horizontal edges

3.4.3 Binary Conversion

The process of converting a color or a gray scale image to a binary format is referred to as thresholding. The method used in this process is a point transformation technique, which creates a binary image from a color or grayscale image on the basis of different pixel intensities. There are several methods for thresholding and this chapter focuses on the fixed threshold method for illustration where the pixel values are assigned the binary value 1 or 0 depending on whether the

pixel values are higher or lower than the intensity threshold value; equations 3.21 and 3.22. The binary value 1 could represent white pixels and binary value 0 could represent black pixels or vice versa [4].

Binary conversion highlights the key details in the image by separating objects in the image or foreground pixels from background pixels to aid in image processing [83], it also reduces the computational complexity. After the two images have gone through the enhancement processes in the pre-processing stage, they are then converted to binary using a thresholding technique.

The input images $f(x, y)$ and $h(x, y)$ have already been gray scaled in the enhancement stages, therefore at this stage the images will just be converted to binary format before image subtraction could take place. In the thresholding process, different values are used to obtain suitable images to be operated on by an XOR circuit, which preforms image subtraction.

Binary conversion operation on an image $f(x, y)$ is expressed as:

for each pixel $p(x, y)$ of image $f(x, y)$

$$p(x, y) = 1 \quad \text{if } p(x, y) > \text{threshold}, T \quad 3.21$$

$$= 0, \quad \text{Otherwise} \quad 3.22$$

3.5 Image Change Detection

After the images have gone through the image enhancement stages, the process that follows is the edge detection. Image enhancement and edge detection processes precede image change detection for the purpose of attaining optimal performance of the algorithm in identifying changes between the input images. When image noise has been reduced, blur removed and edges extracted, the change detection process can easily pick the differences between seemingly identical images. The final process of the change detection framework is contrast stretching which further enhances the features of the output image.

There are several change detection algorithms for image processing and their application depends on the type of images and the fields of application. Some of common generic change detection algorithms applied in image processes to detect changes in digital images are; Image subtraction, background subtraction, Image Log Ratio and the Image Mean Ratio algorithm [84], [85]. In this paper, image subtraction (spot the difference) which is an image processing technique used to detect changes is discussed and then implemented.

3.5.1 Image Subtraction Using Exclusive-OR Operator

The logical exclusive-or (XOR) is a bitwise operator that performs image subtraction between the two image binary data. Image subtraction implementation uses the bitwise logical XOR operator which is a Boolean operator that works on binary data. The binary input images have only two colors (black and white). The bitwise XOR logical operator returns 1 if both bits are different, and returns 0 if both bits are the same, in this experiment considering the pixel value 255 (11111111 binary) representing white pixels and pixel value 0 (00000000 binary) representing black pixels. When the pixel values are the same, the out is (0) and when the pixel values are different, the output is (255), therefore when two pixels are different, the output is a white pixel, and else the output is a black pixel[86] [87].

The XOR bitwise operation preserves any corresponding image pixel pair if the pixels have different values and this operation help to highlight differences in a binary image.

Image subtraction is a process whereby the digital numeric value of one pixel or whole image is subtracted from another image to detect changes between two images. The difference between two images taken at different time is expressed as:

$$g(x, y) = f(x, y) - h(x, y) \quad 3.23$$

where $f(x, y)$ is the first date image, $h(x, y)$ is the second date image and $g(x, y)$ is the difference image [84], [88]. Image subtraction operation takes two images as input and the pixel values of the output image are the pixel values of the first date image $f(x, y)$ minus the corresponding pixels from the second date image $h(x, y)$. Digital image subtraction is performed by an exclusive OR (XOR) logic operator, which is a circuit built from logic gates. The XOR function is only true if just one, and only one of the input values is true, and false otherwise. The XOR operation typically

processes binary images produced through a thresholding method. The logic operation-taking place between the images is a pixel-by-pixel transformation. This bitwise operator manipulates gray-level values at the bit level. The pixel values where the two images differ have a high value (binary 1), representing white pixels and where the two images are the same; the pixel values have a lower value (binary 0), representing black pixels.

Table 3. 1 Truth table for the Exclusive-OR operator

TRUTH TABLE		
Inputs		Output
A	B	C
0	0	0
0	1	1
1	0	1
1	1	0

The XOR operator takes two input images, image $f(x,y)$ and image $h(x,y)$ as shown in design flow in Figure 3.10. It is implemented with four NAND gates (digital logic circuit) [89],[96], a Boolean algebra equation 3.29 and a truth table as shown in respectively.

3.5.2 Exclusive-or (XOR) Operation

The XOR logic operation on two input images is expressed as:

$$g(x,y) = f(x,y) \oplus h(x,y) \quad 3.24$$

Where g is the output image

f is the reference input image (first date image)

h is the input image (second date image)

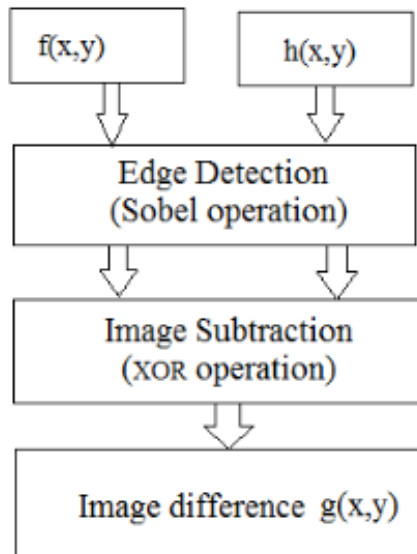


Figure 3.10 Flow diagram for image subtraction using XOR operator

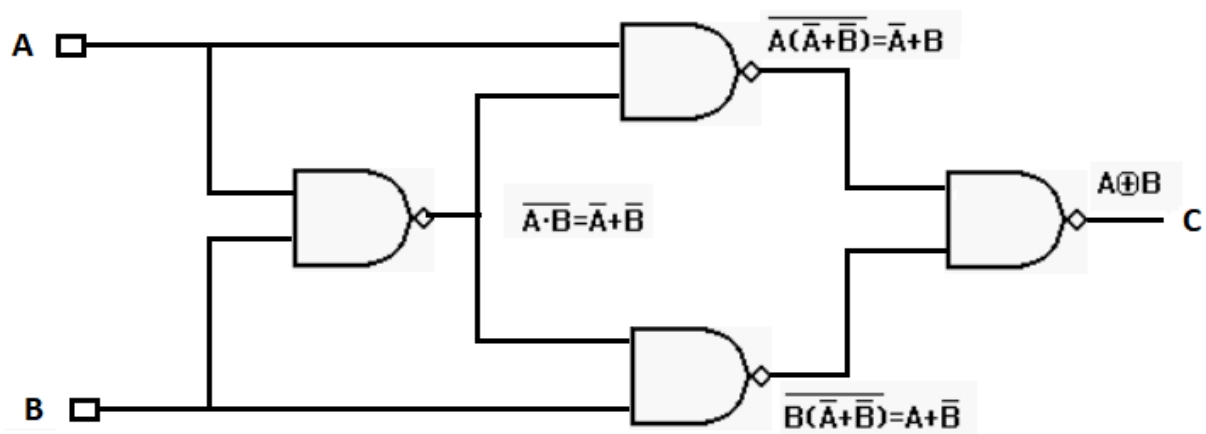


Figure 3.11 XOR gate implemented with four NAND gates

The circuit in represents the exclusive-OR (XOR) operator, which performs image subtraction to produce a difference image C, given two input images A and B. There are four connected logic gates that make up the XOR operator circuit. As an example, the XOR operator circuit takes input image A and input image B and produces a single out, image C depicted in Figure 3.12 (a), (b) and (c) respectively. The mathematical equations for the input and out images can be expressed with a truth table and Boolean equations as shown in equation (3.29).

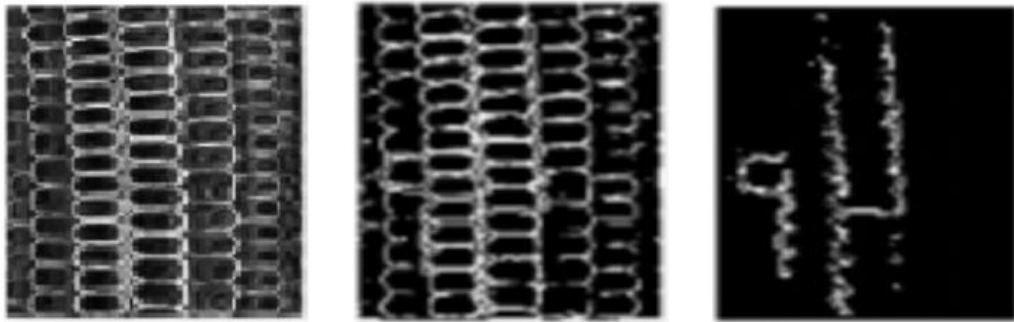


Figure 3.12 (a) Input image A, (b) Input image B (c) output image C (difference image)

3.5.3 Implementations of Boolean Equations

The output equation for the image subtraction is derived from equations of the 2-input, 4-NAND gate XOR operator Figure 3.11 as shown below:

$$\text{NAND gate 1} = \overline{AB} = \bar{A} + \bar{B} \quad 3.25$$

Applying DeMorgan's theorem as derived from Figure 3.11

$$\text{NAND gate 2} = \bar{A} + B \quad 3.26$$

$$\text{NAND gate 3} = A + \bar{B} \quad 3.27$$

$$\text{NAND gate 4} = \bar{A} B + A \bar{B} \quad 3.28$$

$$\text{Therefore output } C = A \oplus B = \bar{A} B + A \bar{B} \quad 3.29$$

3.5.4 Contrast Stretching

The quality of output image is very important, and the contrast enhancement technique is applied in order to enhance this output for improved visual interpretation. Contrast is defined as the difference in the pixel intensity values of pixels which are in close proximity to each other [90]. An image with more contrast appears clearer with more details and an image with less contrast can have a huge impact on human visual interpretation of the image and can lead to less accurate interpretation or representation of the output image.

When the difference in the image intensity is high, then the image is regarded as a high contrast and if the difference in intensity is low, we a low contrast image. High contrast gives better clarity of an image in terms of local details while low contrast gives poor clarity [90].

Contrast stretching is an image enhancement technique that provides a contrast improvement in an image by increasing the range of intensity values it contains to span a desired range of values [91]. In this framework, when the output of the change detection operation is an image with a contrast that is too low for human visual interpretation, the process of image contrast stretching is implemented to enhance its quality. The contrast stretching process is followed by the display stage where the difference image is displayed on an image viewer. Therefore the principal purpose of the contrast stretching operation is to enhance the quality of the image and to reveal more details of the image.

The two commonly used contrast-stretching techniques are the linear and nonlinear. The linear contrast enhancement can be classified into three categories, the min-max linear contrast stretch,

percentage linear contrast stretch and the piece-wise linear contrast stretching [92]. In this paper, the min-max linear contrast stretch will be discussed and demonstrated in FPGA implementation.

3.5.5 Min-Max Linear Contrast Stretching

In this contrast stretching technique new gray level values are assigned to an output image by assigning to the lowest and highest gray level values in the input image values of 0 and 255 respectively. The target gray levels for contrast stretching are 0 and 255 where value 0 is the minimum intensity value and 255 is the maximum intensity value [93].

Linear contrast stretching using min-max mapping can be expressed as

$$g(x,y) = \left(\frac{(f(x,y) - \min)}{(\max - \min)} \right) \times 255 \quad 3.30$$

where $f(x,y)$ is input image.

$g(x,y)$ is the transformation function for the image, that produces the gray level of the output image from the contrast stretching process. The gray level range of the output image is determined by the "min" and "max", which represent the minimum intensity value and the maximum intensity value in the current image respectively [94].

In an 8-bit gray scale image, the linear contrast stretching process maps the minimum gray level value to 0 and the maximum gray level value to 255 with the rest of the gray level values remapped linearly between 0 and 255 to generate a high contrast image. The output image (high-contrast image) has intensity values that extend across the whole image gray level spectrum, Figure 3.14 depicts a contrast stretching of the image Lena Figure 3.13, [95].



Figure 3.13 Original image (low contrast)



Figure 3.14 Image with high contrast after contrast stretching

3.6 Design and Modeling of the Framework

The first step in the development of the FPGA-based change detection framework was the framework design, which included building models, creating diagrams, writing source codes and creating schematics. The schematics were designed on the Xilinx Integrated Software Environment (ISE). The source codes were created in MATLAB and in VHDL, a Hardware Description Language. The model-based designs were created in the Mathworks environment using both the Xilinx system generator (XSG) and Simulink platforms. The design stage was followed by the design verification (functional simulation). This is the stage where the validation that the functionality of the design blocks complies with design block requirements is performed.

For targeting Xilinx FPGAs, the XSG engine performs cycle accurate and bit precise simulations and generates synthesizable files from the model-based designs that serve as input to the FPGA implementation.

In this thesis, the simulation was performed in the Xilinx ISE and MATLAB/Simulink programs. The design synthesis of the Mathworks model-based image change detection design was performed in the integrated software environment of the Xilinx system generator and Simulink platforms, Figure 3.15. This process was automatically executed specifically by the system generator unit. The entire Simulink/ XSG model-based design of framework implementation is composed of the following three phases; Image Pre-Processing Blocks unit, Xilinx system generator unit, and the Image Post Processing Blocks unit.

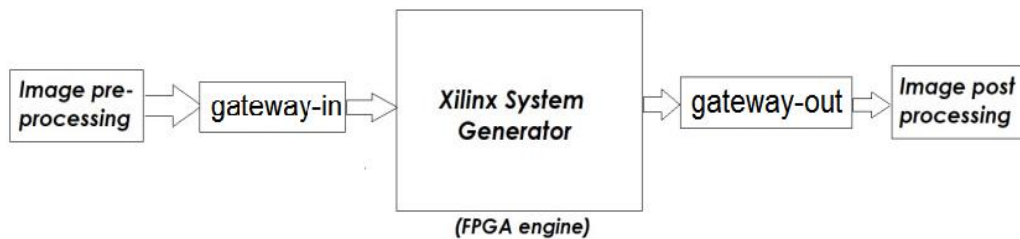


Figure 3.15 Simulink/Xilinx system generator design flow



Figure 3.16 1st date Original image

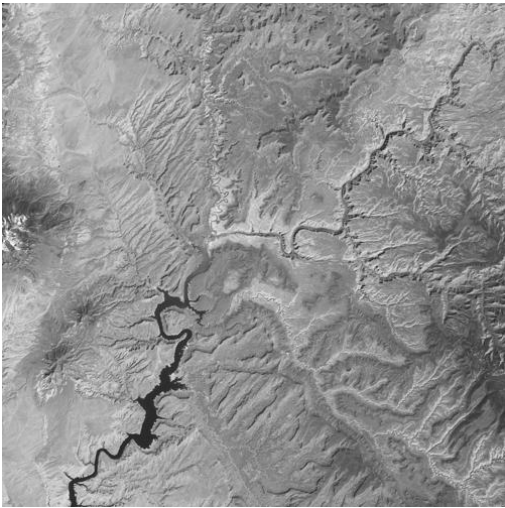


Figure 3.17 1st date image (gray scale)

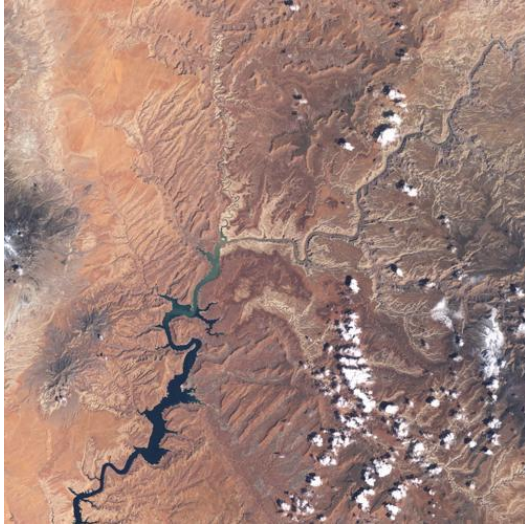


Figure 3.18 2nd date Original image

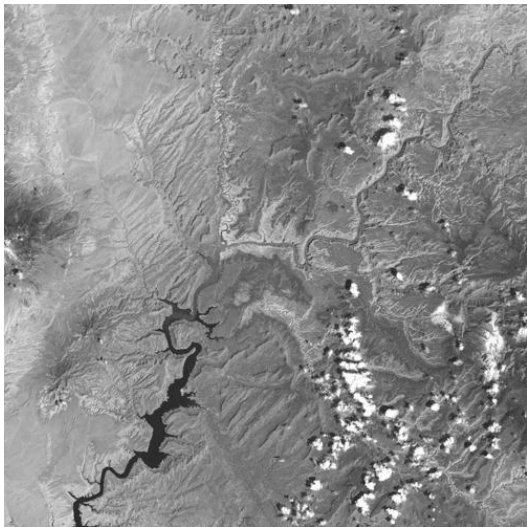


Figure 3.19 2nd date image (gray scale)

3.6.1 System Implementation

This section discusses and describes the implementation of the proposed FPGA-based change detection framework that guides the image change detection process using FPGA system. The implementation method presented in this chapter is the concept of hardware/software co-simulation for image processing using high-level prototyping software tools; MATLAB/ SIMULINK, Xilinx Integrated Software Environment (ISE) synthesis platform, and finally the Xilinx FPGAs system. The hardware implementation of the image processing algorithms targeting the FPGA architecture is done using model based block design approach. The implementation of image enhancements processing techniques was performed using the Xilinx system generator and the MATLAB/Simulink environment. The Simulink HDL coder was used to generate VHDL codes for targeting implementation in an FPGA. The design flow depicted in Figure 3.20, shows the image pre-processing stages of the framework. The edge detection and image change detection algorithms were then performed using VHDL for both simulation and synthesis. This procedure creates a code in the form of bit stream, suited for hardware implementation. The bit stream could then be executed in the FPGA. The framework design flow depicted in, shows image processing stages for the FPGA-based implementation of the framework.

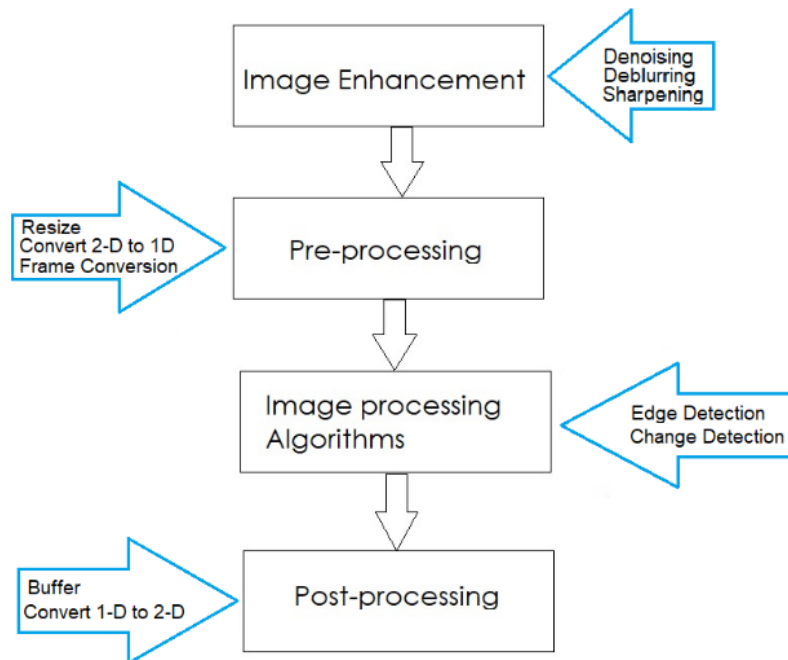


Figure 3.20 Design flow for Framework Implementation

The inputs to the framework implementation process are the two images Figure 3.16 and Figure 3.18. These two similar images were obtained from the NASA observatory data set [96]. The images were converted to a gray scale format (Figure 3.17 and Figure 3.19 respectively), which is a suitable format for the implementation process. These images are of exactly the same region and were taken at different time periods. The latest image Figure 3.18 was acquired on April 13, 2003. A river with some water, small streams and few patches of snow can be noticed in the image. The first image, Figure 3.16 was captured on May 12, 2002. The most prominent parts of this image are the river, the level of water and the patches of more snow on the ground. These images are suitable for application in this research and used in implementing image change detection in the proposed framework to find the differences between the two images.

3.6.2 MATLAB/Simulink

It is a mathematics software that is used for manipulating matrices and visualizing data. Within MATLAB, there exists the image processing toolkit, which provides extra functionality for specialized applications that include image enhancements functions such as image filtering and feature detector operations. Simulink is a model-based design environment for modeling, simulating and analyzing designs. Simulink has customizable set of block libraries and can be integrated with the Xilinx system generator environment. The block sets are compatible with the system generator blocks and interconnect to form a single block-based design.

3.6.3 Xilinx System Generator (XSG)

The Xilinx system generator is a highly optimized Digital Signal Processing (DSP) library that enables the creation of Mathworks model-based design, VHDL codes, and MATLAB derived functions. These designs can then be simulated within the Simulink/XSG environment and then compiled for FPGA implementation. The XSG contains a library of Simulink blocks, memories, DSP functions. The XSG also automatically generates VHDL or verilog code from the block model that can be synthesized and be implemented in Xilinx FPGAs [86].

The system generator comes with a predefined block set along with MATLAB/SIMULINK software package that is used for implementing algorithms for FPGA design. The gateway blocks

(gateway-in, and gateway-out), which are the basic elements of the Xilinx block set, provide the connection between the Xilinx system generator and Simulink blocks. The purpose of the gateway-in is to convert the Simulink input of type format (integer, single or double) to the Xilinx data type. The gateway-out converts the Xilinx data type format (fixed point or floating point) into Simulink data type format.

The system generator also consists of a VHDL code generator, which automatically generates hardware description language (HDL) code for models within the program. In this thesis, the implementation of the last two stages of the proposed framework, the system generator was used to create models for edge detection and change detection. One of the most important features of the XSG is the bit and cycle accurate floating and fixed-point implementation [86]. This phenomenon implies that the output data that is observed in the simulation precisely matches the output data that is generated in the FPGA hardware [87]. The framework implementation diagram Figure 3.15, shows the connection of the pre-processing unit, the XSG (FPGA engine) and the post-processing unit.

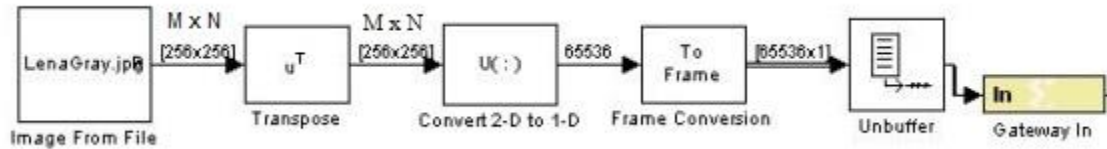


Figure 3.21 Image Pre-Processing Blocks unit

3.6.4 Pre-processing for FPGA Implementation Using XSG

The input to the block-based design change detection framework is a pair of gray scale image, Figure 3.17 and that are pushed through the blocks of the pre-processing unit, Figure 3.21. The first entry is through the transpose block, which transposes $M \times N$ matrix to $N \times M$ matrix form. Then the Convert 2-D ($N \times M$) to 1-D block converts the input data that is in two dimensional to one-dimensional format. The Frame conversion block sets the output signal to frame based set the

one-dimensional signal to a frame-by-frame based data. The unbuffer block converts this frame-based data to digital samples at a higher sampling rate. The block is then connected to the gateway-In block where Xilinx system generator implementation.

3.6.5 Post Processing for Image Enhancement Processes

The post processing block unit connects to the Gateway-out, which is the output block of the system generator, Figure 3.22. The post-processing unit recreates the output from the FPGA and restores it to a 2-D format data. The output at Gateway-out is a floating point and it has to be converted to unsigned integer of 8-bit. The first block in this unit is the buffer block, which converts digital samples to frame-by-frame data at a lower sampling rate. The data then goes to through the Convert 1-D to 2-D block, which convert the data to one-dimensional format. The output is then passed through the transpose block, which translates the data to back to MxN matrix. At this stage, the output image can be displayed on an image viewer.

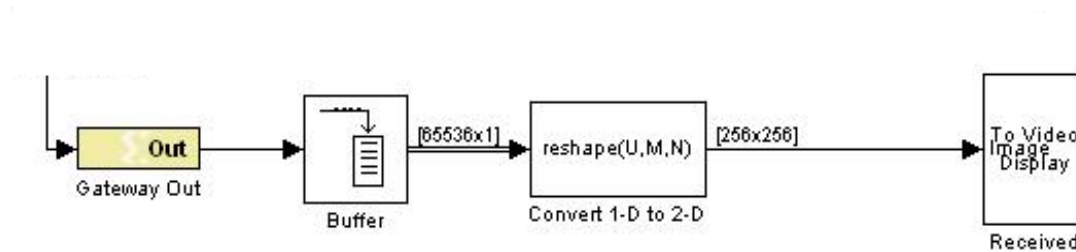


Figure 3.22 Image Post-Processing Blocks unit

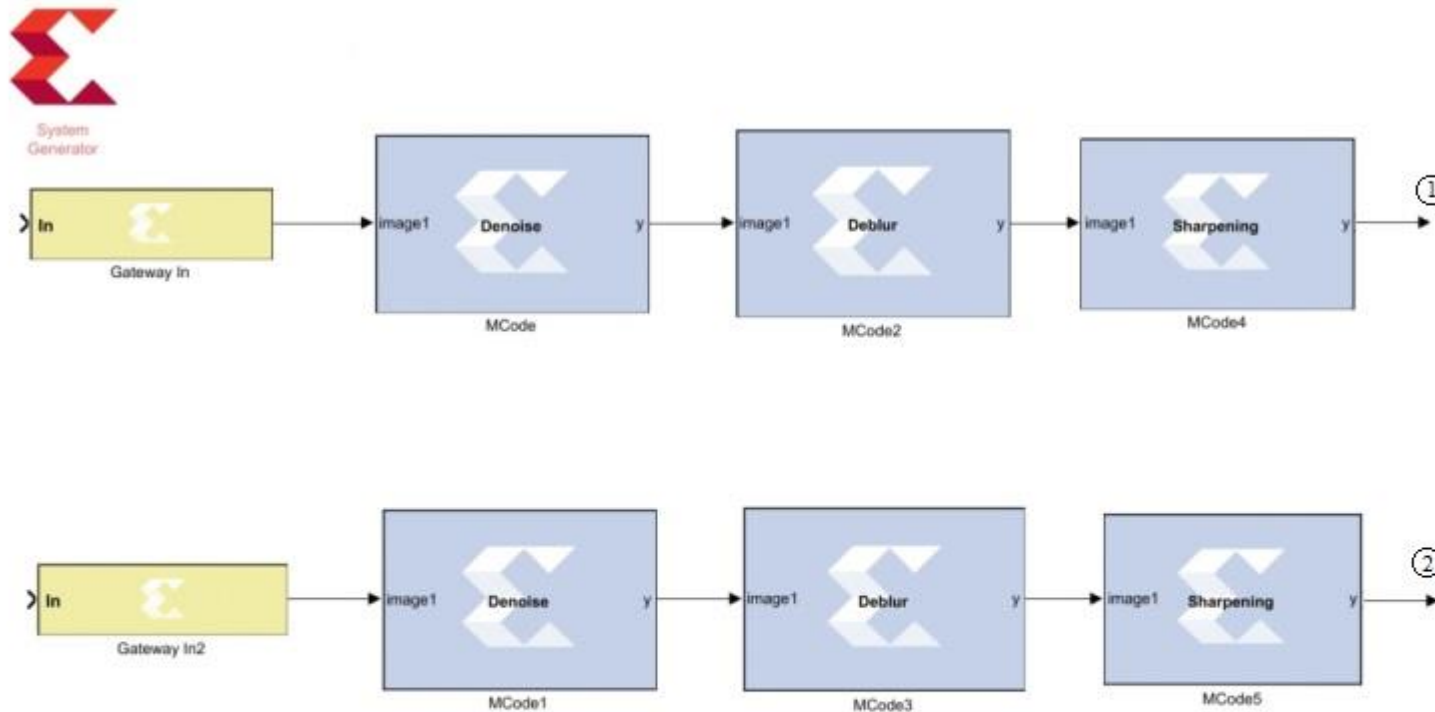
3.6.6 Framework Design and Experimental Setup

The change detection framework was designed within a design environment whose main components are the Mathworks MATLAB, Simulink, Xilinx system generator and Xilinx ISE. The framework design in this thesis is a model-based design that utilizes the Simulink and Xilinx block sets to build the different image processing operation modules that make up a complete model-based change detection framework. This design and experimental setup is a hardware/software co-simulation developed within the MATLAB/Simulink modeling and simulation environment, and the synthesis phase is employed on FPGA hardware using VHDL. Various Xilinx and Simulink

block sets; including the pre-processing block unit and the post processing block unit connect together to form a complete change detection framework.

The model based block diagram in Figure 3.23 shows image enhancement and restoration phase of the framework, while the block sets for edge detection and change detection operations are shown in Figure 3. 24.

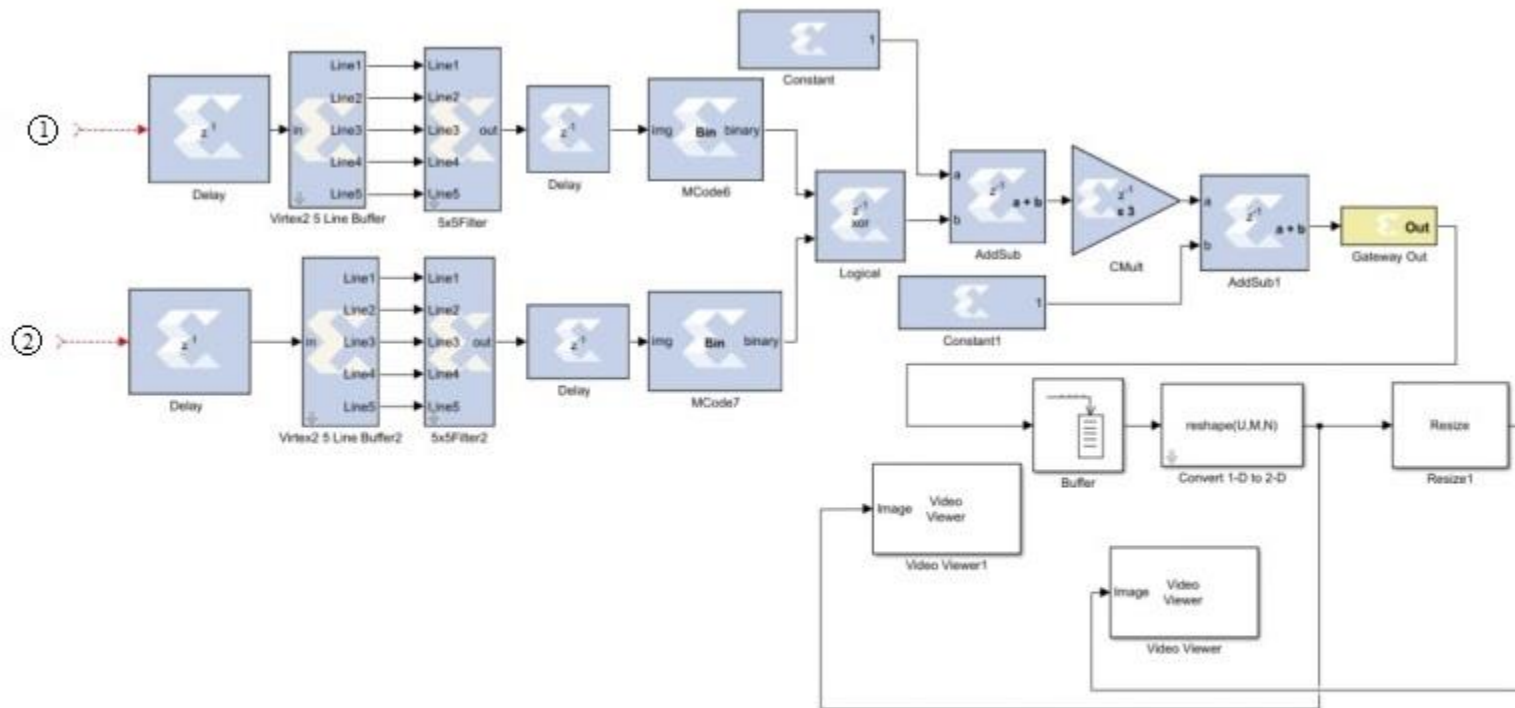
The enhancement and restoration unit comprises two sets of the gateway-in block, denoising, deblurring and the sharpening blocks. Each set of this unit performs image enhancement and restoration processes to the input images 1st date image and 2nd date image. The image output of each of the sub units is passed to the image edge detection phase for further processing. The functions of the block sets in these diagrams are explained in the framework implementation section 3.7.



key:

- ① Connects to the delay block of the next phase (Edge detection and Image subtraction block unit
- ② Connects to the delay block of the next phase (Edge detection and Image subtraction block unit

Figure 3.23 Image enhancement and restoration phase



Key :

- ① connects to the image sharpening block unit of the image enhancement and restoration unit
- ② connects to the image sharpening block unit of the image enhancement and restoration unit

Figure 3. 24 Image change detection phase

3.7 Framework Implementation

3.7.0 Overview

The implementation of the framework involves the execution of all the image processing operations corresponding to the Simulink and Xilinx block sets that make up the whole change detection framework. The implementation follows the sequential order as described in the design flow, Figure 3.20. The image enhancement and restoration operations form the first phase, which is then followed by the edge detection operation. The last phase is the image subtraction operation, which executes the change detection and the contrast stretching. This chapter answers the research questions (i), (ii) and (iv)

3.7.1 Implementation of the Image Change Detection Framework

The framework design was compiled using a hardware co-simulation compilation driven by the system generator token. The generated bit file from the hardware co-simulation compilation got transferred to the FPGA. The framework components produce output data from individual components and this output data is pushed as input data to the next image processing operation. This framework model-based design was implemented and verified using Xilinx system generator integrated with the MATLAB/Simulink design environment.

The principal goal of this framework is to generate an image difference between two images, by detection changes that exist between them. The pair of gray scale images Figure 3.17 and Figure 3.19 (courtesy of NASA) [84], [85], acquired at different time periods were used as the input images to the change detection framework. The detection of changes between two images involves several processes. This image change detection framework implementation addresses the research questions (i) and (ii) listed in Chapter 1.

- Preparing input images for framework implementation

The implementation of the image change detection framework was performed using the images from the NASA observatory data set. Since these images do not show any obvious image blur or noise, some impurities were added to these input images. Salt and pepper noise was introduced in the image data to create random black and white pixels. Image blur was added to the input image data to create a blurry effect. The images were then transformed to gray scale,

and this procedure was used to check whether the image processing techniques that follow would produce the results as expected. A noise free and blur free output images were expected after the each denoising and deblurring stage respectively.

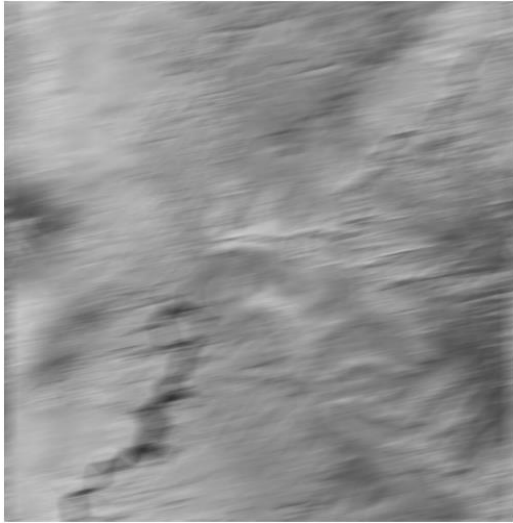


Figure 3.25 1st date image (blurry and noisy)

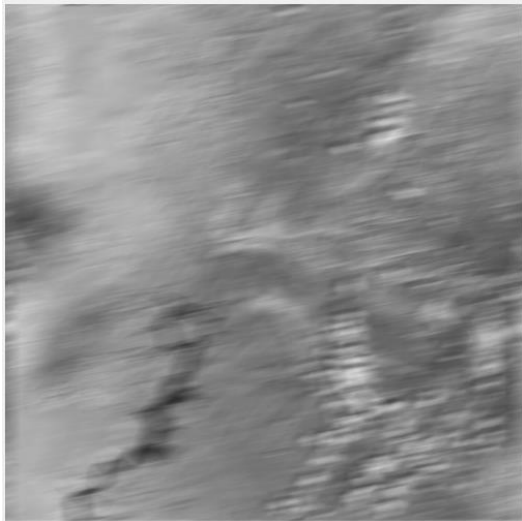


Figure 3.26 2nd date image (blurry and noisy)

The blurred and noisy input images are shown in Figure 3.25 and Figure 3.26. The black and white spots that depict the presence of salt and pepper noise on the surfaces of the image in Figure 3.25 and the image in Figure 3.26 are not visible to the naked eye because the images are blurred.

- Image enhancement and restoration phase

The image enhancement and restoration operations form the initial stage, where denoising, deblurring and image sharpening take place in this sequential order. The first step is noise removal, which is performed by a denoising algorithm embedded in the denoising block set. The image data is then pushed to the deblurring operation block set which is the next process in the framework. This operation performs image blur removal. The image data is now pushed to the image-sharpening phase of the enhancement and restoration stage where fine details of the images are extracted. The image data then gets pushed to the next stage in the framework, which is binary image conversion.

- Binary conversion and edge detection phase

In this phase of the framework, the gray-scaled images which are noise and blur free at this stage are converted to binary format using a thresholding operation in Simulink block. The output of the binary image conversion module is pushed to the edge detection MATLAB/Simulink block set. This edge detection model performs edge extraction on the image data and pushes the data to the image subtraction phase.

- Image subtraction phase

After edge detection operation, the next stage is the image subtraction operation that makes the core process of this framework. The image subtraction block which is directly linked to the two image data outputs block set of the edge detection module via the Xilinx gateway-out block performs a bit-wise change detection operation on the image data. The image subtraction block performs change detection using the Xilinx exclusive-OR logical. This operation produces an image difference and this output is pushed to the contrast stretching block module, the final image processing stage of the framework.

- Contrast stretching phase

The phase of the frameworks comes at the later stage and its main purpose is to improve the contrast of an image by increasing the dynamic range of intensity pixel values, causing the image to utilize entire range of values available to it. The image contrast stretching block unit, which immediately follows the image subtraction unit performs a stretching operation on image data (difference image). The image difference that has a low contrast is then transformed into a high-contrast image by remapping or extending the pixel values. This contrast stretching operation produces an image that has high contrast with pronounced features that reveal the fine details of the image.

3.7.2 FPGA Implementation of Image Denoising

The Simulink/XSG model-based design for noise removal comprises mainly the median filter (Mcode Block), the Xilinx gateway-in and gateway-out blocks as shown in Figure 3.29. The noisy and blurry images Figure 3.27Figure 3.27 and Figure 3.28Figure 3.28 were fed as inputs to the denoising blocks where they went through the noise removal process to produce noise free image data. The median filter function through the Mcode block performed a noise removal process by calling the denoising M-file, Listing 2. The outputs went through the post processing blocks where they got transformed back to 2-D images with salt and pepper noise removed. The salt and pepper noise is the only impurity removed at this stage. The image blur is removed in the deblurring stage.

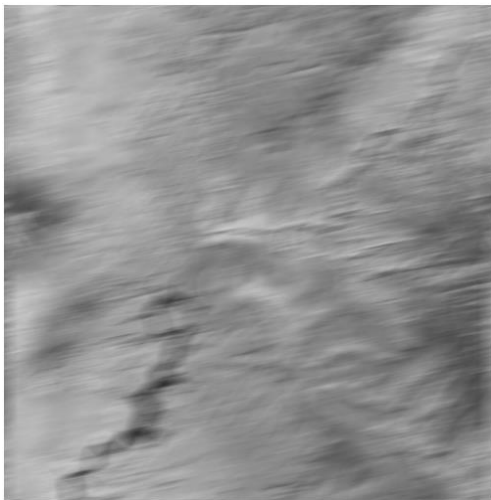


Figure 3.27 1st date image (noisy and blurry)

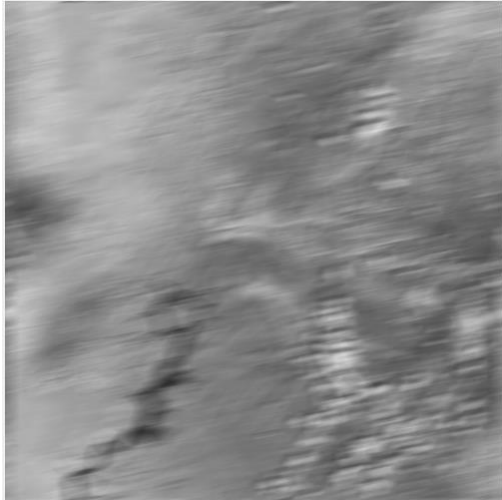


Figure 3.28 2nd date image (noisy and blurry)

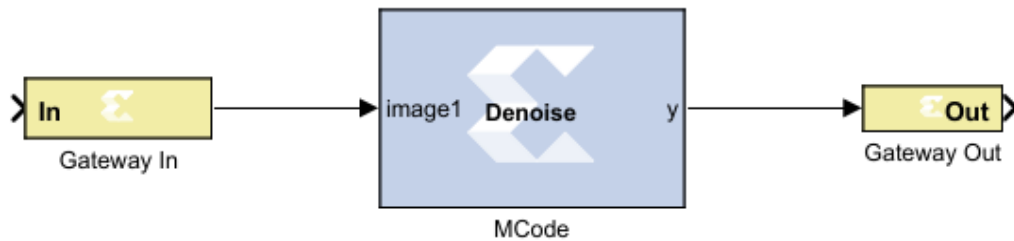


Figure 3.29 Image noise removal Xilinx/Simulink model design

3.7.3 FPGA Implementation of Image Deblurring

Image deblurring implementation on Simulink/XSG environment is performed using model-based design that is made up of the Xilinx gateway-in, gateway-out block sets and the denoising Mcode Block, Figure 3.32. The Mcode block implements the deblurring algorithm by calling the deblurring M-file, Listing 3 which uses the Wiener filter for removing the blur. The input image data is passed to the pre-processing block, here the data got converted into a one-dimensional array and was

passed into the Xilinx gateway-In block, then to the Xilinx Mcode block for blur removal. At this stage the image data was blur free and was then pushed through the post processing stage where was it transformed back to 2-D dimension.

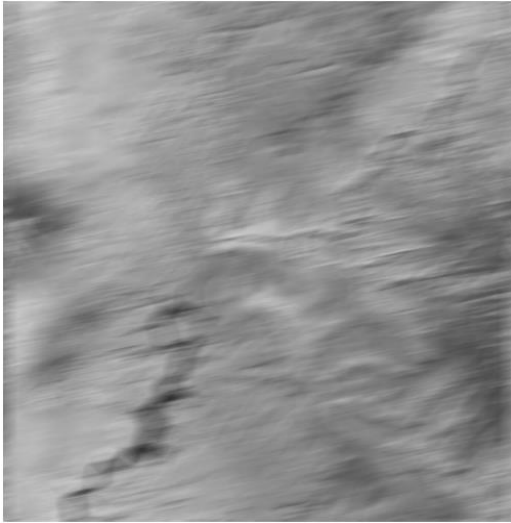


Figure 3.30 1st date image (blurry, no noise)

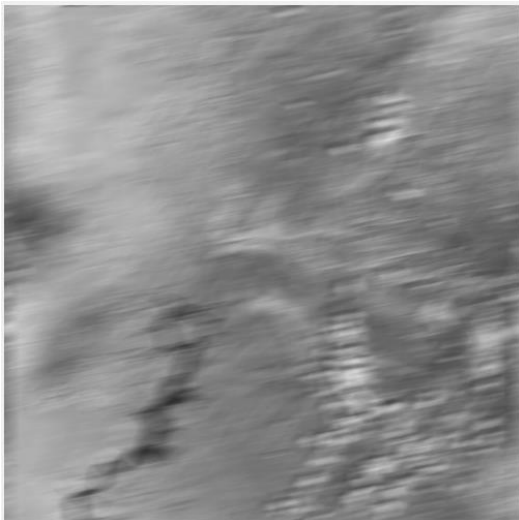


Figure 3.31 2nd date image (blurry, no noise)

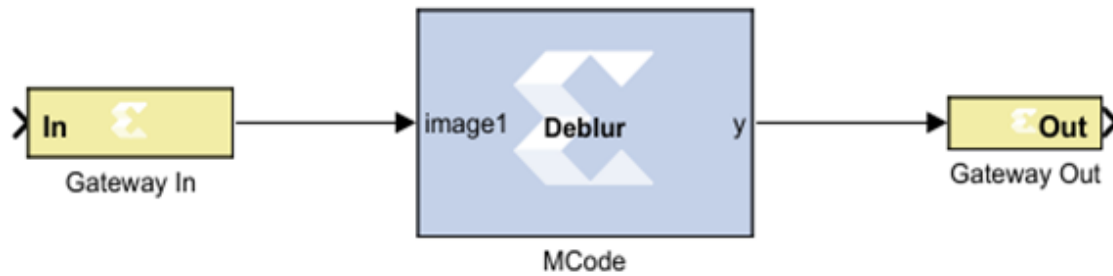


Figure 3.32 Image blur removal using Xilinx/Simulink model design

3.7.4 FPGA Implementation of Image Sharpening Using XSG

Image sharpening was implemented using Simulink and Xilinx system generator model-based design. The block design comprises the Xilinx gateway-in and gateway-out block-sets and the Mcode Block.

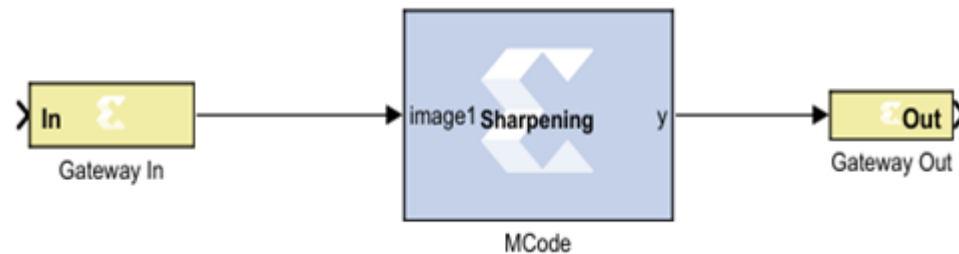


Figure 3.33 Image sharpening using Xilinx/Simulink model design

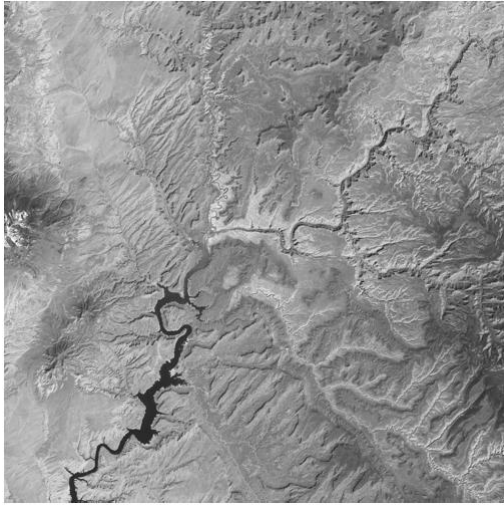


Figure 3.34 1st date image (blur free and noise free)

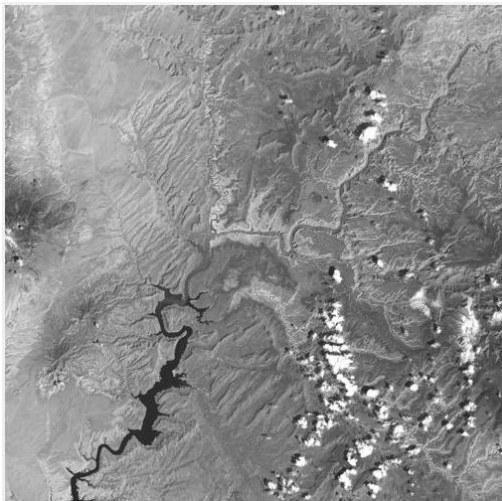


Figure 3.35 2nd date image (blur free and noise free)

The output images from the deblurring stage, Figure 3.34 and Figure 3.35 were taken as inputs to the image sharpening phase. The Xilinx Mcode Block in Figure 3.33 performed image sharpening process by calling the unsharp-masking filter embedded M-file, Listing 4. The images were pushed to the Xilinx output (gateway-out) and then passed to the post processing blocks where they got transformed to 2-D images with crispy edges and fine details.

3.7.5 FPGA Implementation of the Sobel Edge Detection Using XSG

The edge detection operation was performed in the images to extract edges so that the change detection process (next stage in the framework) can be more effective in revealing the changes between the images. Then the images were taken as input to the edge detection operation. These two grayscale images have passed through the image enhancement and restoration stages of the framework, and they are all denoised, deblurred and sharpened.

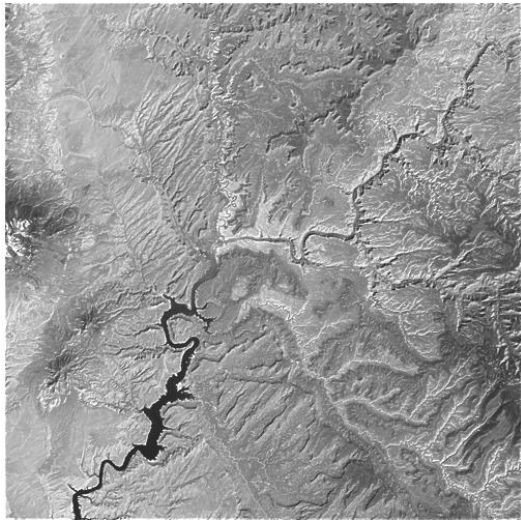


Figure 3.36 1st date image (sharpened)

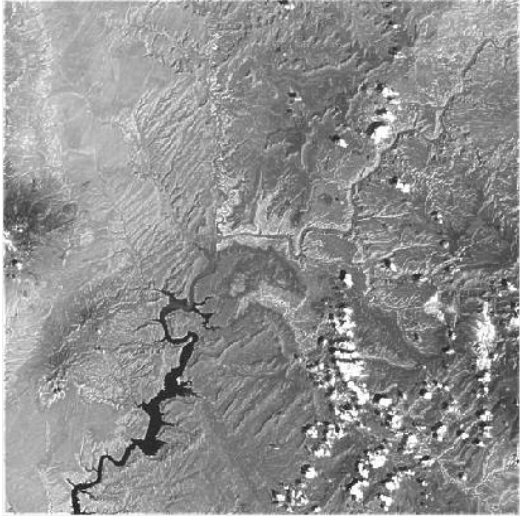


Figure 3.37 2nd date image (sharpened)

The Sobel edge detection operation was implemented using Simulink and Xilinx system generator model-based design, Figure 3.38. This model-based block design unit consists of the Virtex2-5 Line Buffer, 5x5 filter, Delay and the binary converting block: The Xilinx Virtex2-5 Line Buffer block is made up of four virtex2 line buffers all connected in series. This block buffers a sequential stream of pixels, constructing five channels of output data. Each channel is delayed a specified buffer depth (N) samples. Line 1 is delayed 4*N samples, each of the following lines are delay by N fewer samples, and line 5 is a copy of the input data. The Virtex2-5 Line Buffer connects to the Xilinx 5x5 Filter block that uses nine different 2-D FIR filters; the Edge, Smooth, Sharpen, SobelX, Sobel Y, Sobel-XY, Blur, Gaussian and the Identity.

In this edge detection implementation, the Sobel-XY filter was used for edge detection which implemented convolution operation with input pixel values in the horizontal (k_y) and vertical direction (k_x).

The first image, 1st date image, Figure 3.16 was fed as input through the Simulink pre-processing blocks where it went through the pre-processing stages to produce an appropriate input to the gateway-In. The model-based block design unit connected between the Xilinx gateway-In (input to FPGA) and the gateway-out (output of FPGA) performed the Sobel edge detection on the image.

The output data via, Xilinx gateway-out went through the Simulink post processing block unit where the image was transformed back to a 2-D image with pronounced edges. The second image data, 2nd date image Figure 3.18 went through the same process of edge detection. The image output data Figure 3.36 and Figure 3.37 Figure 3.37 from the Edge detection unit is passed to the binary conversion unit, which converts the gray scale image data to a binary format. The image data was then fed to the image subtraction phase (logical XOR block) that performed change detection between the image data from 1st date image and 2nd date image.

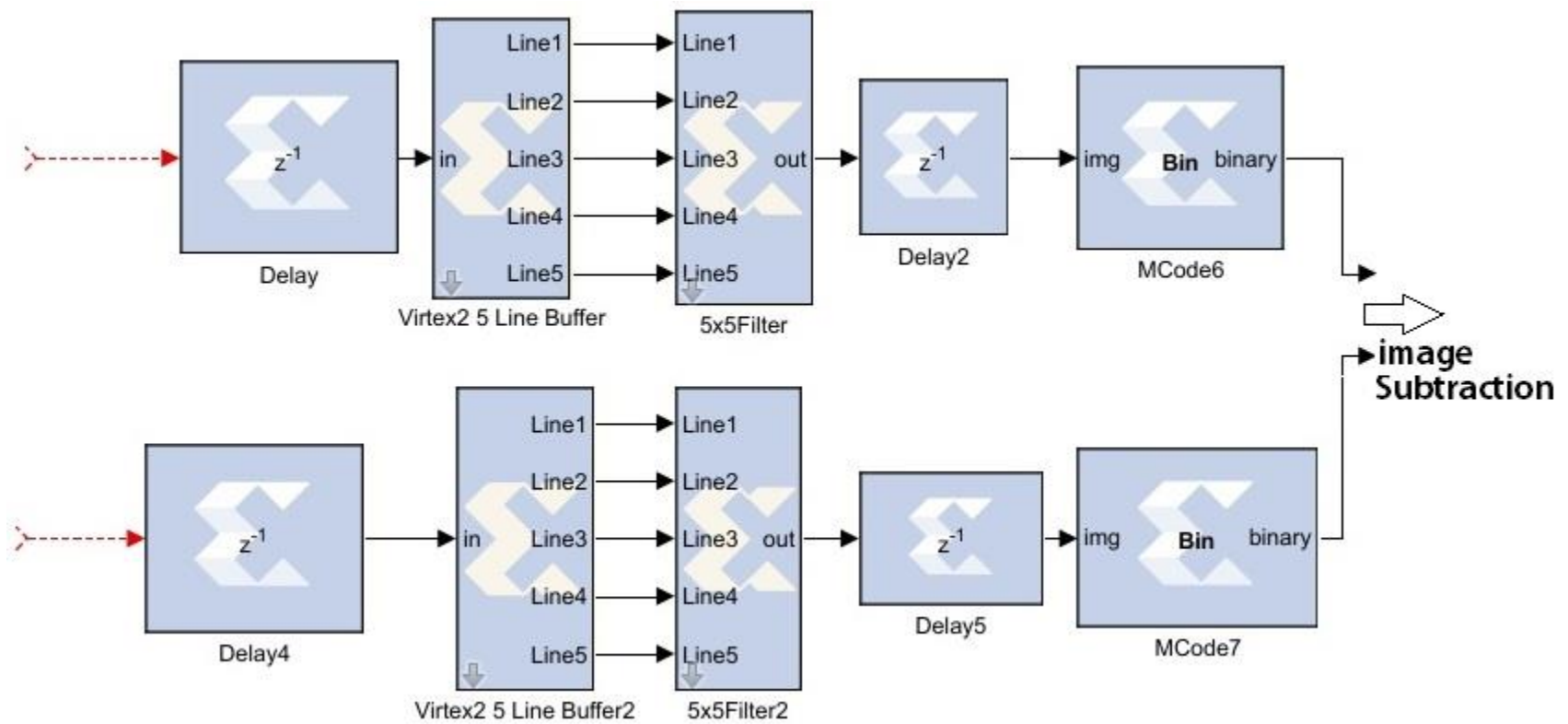


Figure 3.38 Image Edge Detection using Sobel Operator

3.7.6 Implementation of Change Detection Using ISE

Change detection between two images was implemented using image subtraction, a process whereby the digital numeric value of one pixel or whole image is subtracted from another image to detect changes between two images [97]. This process as explained in Chapter 3.6 is carried out by the exclusive-OR (XOR) operator, which performs image subtraction to produce a difference image C, given two input images A and B. The XOR operator schematic for image subtraction Figure 3. 39, was designed and simulated using Xilinx Integrated Software Environment (ISE) and VHDL code, Listing 1. The waveform in Figure 3. 40 shows the output from ISE simulation.

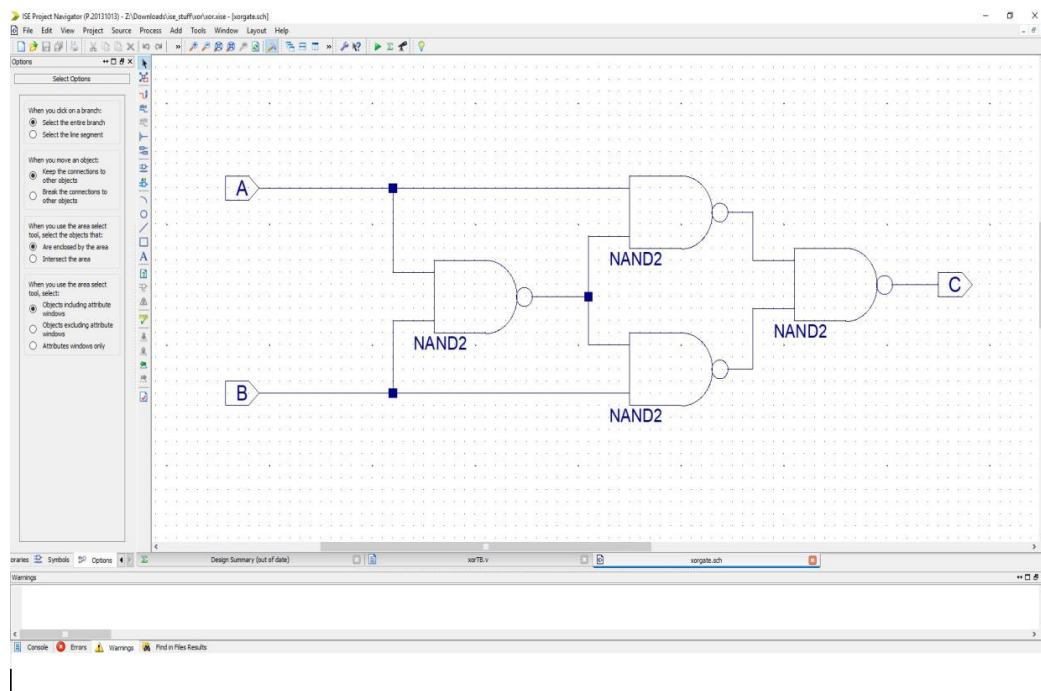


Figure 3. 39 Schematic for image subtraction using XOR operator



Figure 3. 40 Simulation of image subtraction using XOR operator

3.7.7 FPGA Implementation of Image Change Detection Using XSG

The change detection process was implemented using Simulink and Xilinx system generator model-based design. The two images Figure 3. 41 and Figure 3. 42 show extracted edges and they are the output images from the preceding framework phase (Sobel edge detection stage).

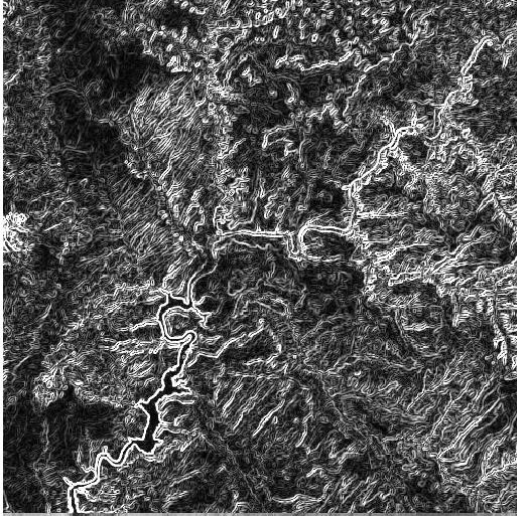


Figure 3. 41 1st date image (edges detected)

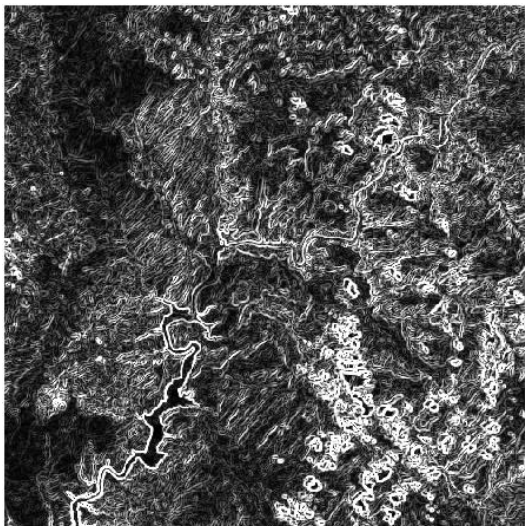


Figure 3. 42 2nd date image (edges detected)

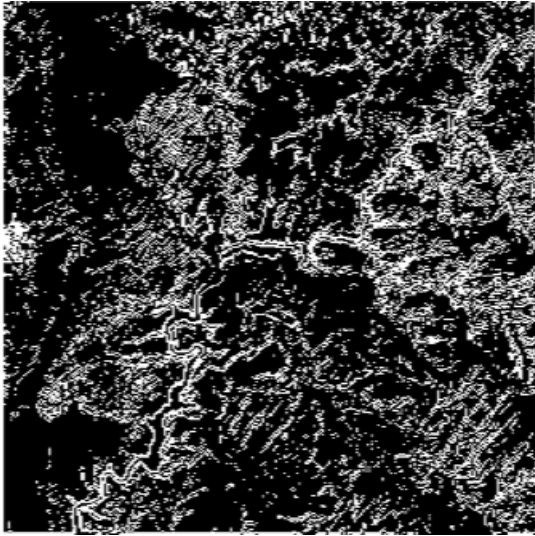


Figure 3. 43 Binary 1st date image

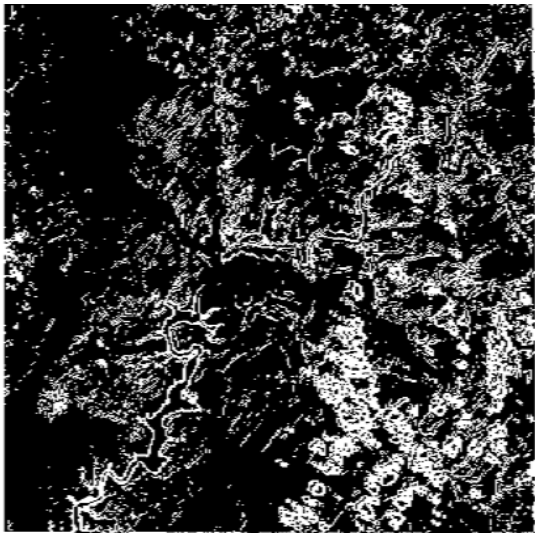


Figure 3. 44 Binary 2nd date image

These two images were pushed as inputs through the Simulink pre-processing block units, each image went through its own pre-processing unit via a Xilinx Gateway-In block. In the Simulink pre-processing unit, the input images Figure 3. 41 and Figure 3. 42 were converted to binary images Figure 3. 43Figure 3. 43 and Figure 3. 44 respectively, a format that is suitable for the FPGA logical unit. The FPGA engine in this case is the Xilinx logical block (XOR gate unit). The two input blocks (Gateway-In, and Gateway-In2) connect directly to the XOR logical model block.

The FPGA engine received two binary image inputs images 1st date image and 2nd date image, performed an image subtraction operation thus extracting the differences between the input binary image Figure 3. 43 and binary image Figure 3. 44Figure 3. 44. The output of this unit via the gateway-out, was passed to the Simulink post processing unit that transformed the difference image to a 2-D output.

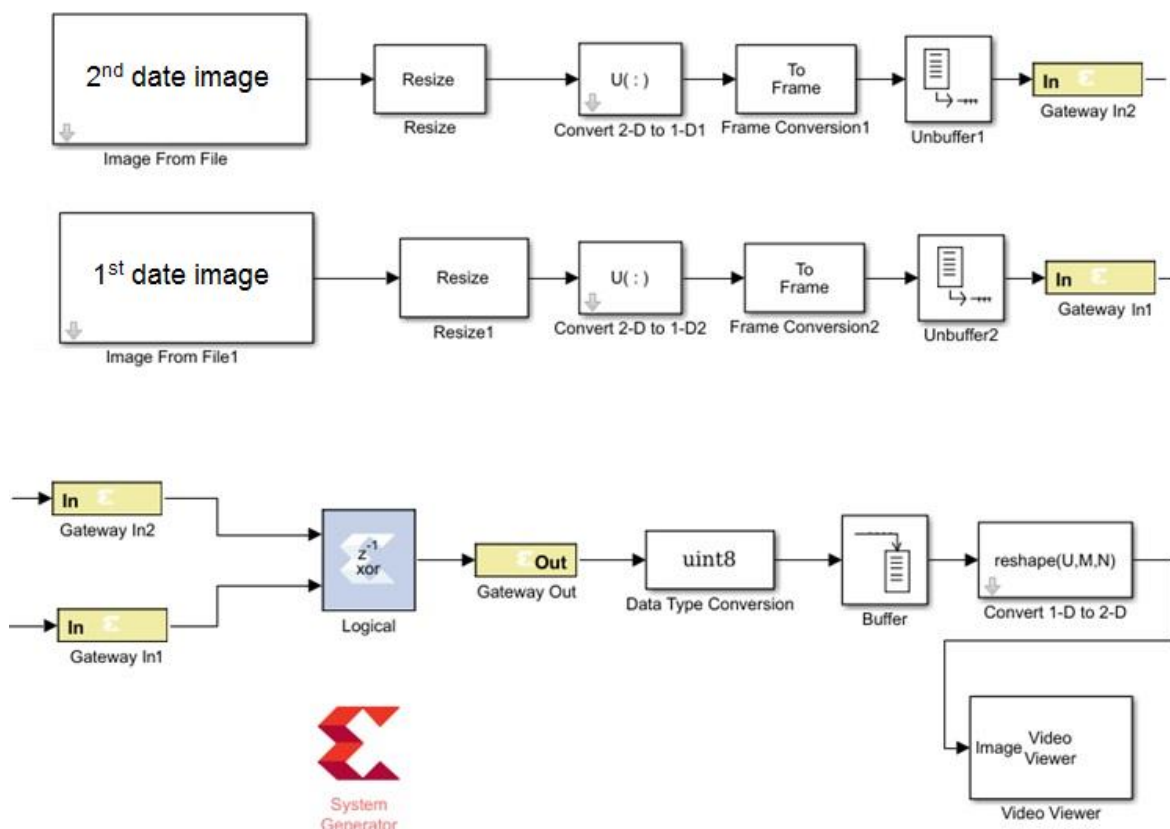


Figure 3.45 Design flow for image change detection implementation

3.7.8 FPGA Implementation of Contrast Stretching

Contrast stretching was implemented using Simulink and Xilinx system generator model-based design shown Figure 3. 46. The block design is built from the Simulink pre-processing, post processing block-sets and the Xilinx contrast block whose purpose is to perform image contrast stretching. The output image from the change detection processing stage was fed as input through the pre-processing block design to produce suitable input to the gateway-In. The Xilinx contrast block which takes in a single input connects to the input and output blocks. This block performed image contrast stretching process by calling the linear contrast stretching m-file. The image data was then pushed to the post processing blocks via gateway-out where it got transformed to a 2-D image. The expected output is an image that has enhanced quality for improved visual interpretation.

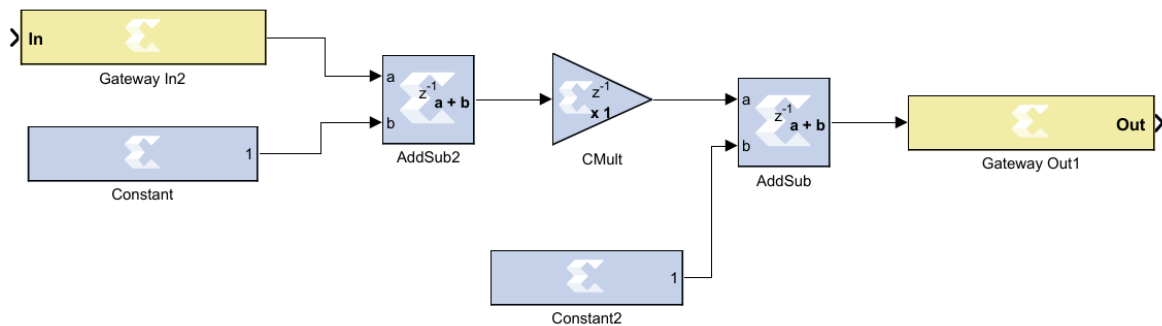


Figure 3. 46 Contrast stretching block design

Chapter 4: Discussion and Analysis of Results

4.0 Overview

Chapter 4 presents discussion and analysis of experimental results of the modelled FPGA-based change detection framework with reference to the objective of the research study, which was to design and develop a framework for FPGA-based change detection. The discussion is in two parts, the results from the individual modules of the framework and the ultimate results of the framework as a whole.

4.1 Results from Components of the Proposed Framework

4.1.0 Overview

The implementation of the individual components of the change detection framework was performed in the sequential order as described in Chapter 1. This procedure was critical in providing high quality input data to the image subtraction detection phase, which is the core operation of the change detection framework. The discussion of results described in this section of the thesis comes from every stage of the framework and they show vital role played by every stage of the framework.

4.1.1 Results for Image Denoising

The aim of the implementation of noise removal was to get rid of the salt and pepper noise (white and black spots) in the images. The 1st date and the 2nd date image had salt and pepper noise added to them but the black and white spots that indicate the presence of image noise are not visible due to the image blur present in the images.

The Xilinx/Simulink design in Figure 3.29 executed denoising process using median filtering to remove the salt and pepper noise from images Figure 4. 1 Figure 4. 1 1st date image (noisy

& blurred image) and Figure 4. 3, which are now represented below as Figure 4. 2 and Figure 4. 4 respectively.

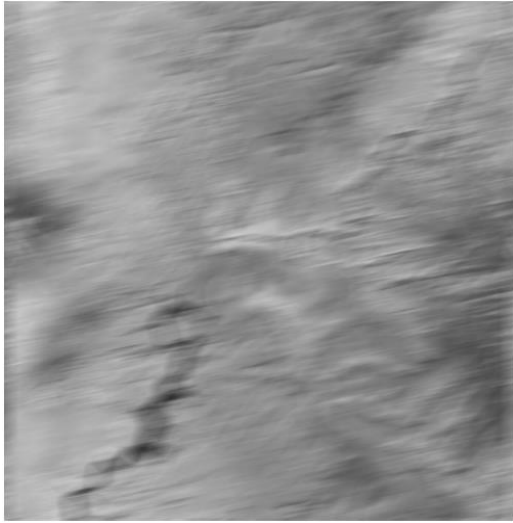


Figure 4. 1 1st date image (noisy & blurred image)

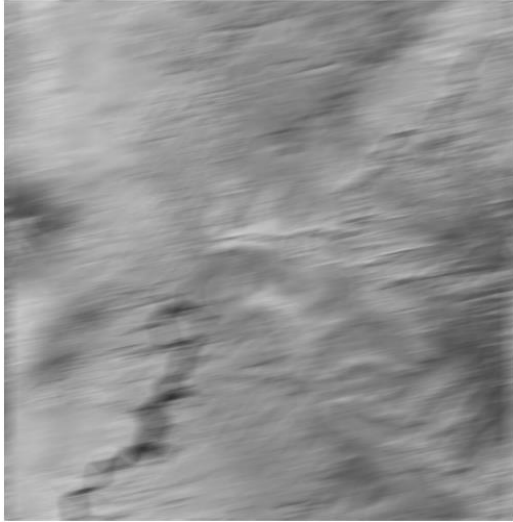


Figure 4. 2 1st date image restored image (noise removed)

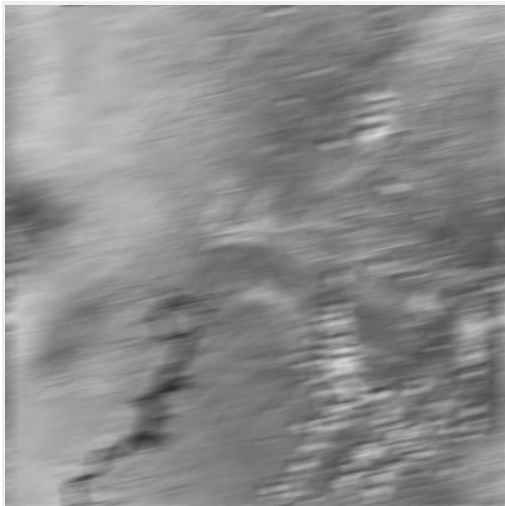


Figure 4. 3 2nd date image (noisy & blurred image)

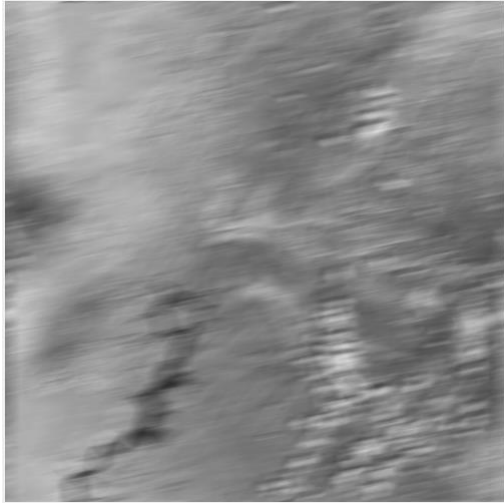


Figure 4. 4 2nd date image (noise removed)

The restored images Figure 4. 2 and Figure 4. 4 are the output images that are noise free. The image noise on both images have all been removed by the median filter, but the image blur that still exists in the images and makes it difficult for the naked eye to pick the difference between the input and the output images.

The experimental results from the denoising model design show that the implementation using the Xilinx system generator has performed and generated the expected outcome, hence validating the denoising algorithm. The output of this phase of image restoration indicates that the median filtering process is very effective in removing salt and pepper noise. This phenomenon confirms the critical need for placement and inclusion of this operation in the image pre-processing phase of the change detection framework. It can then be concluded that this phase of image restoration is necessary and indispensable in the framework, otherwise without it the noise in the input images would be carried to the next phase of the framework and end up compromising the quality of the next output image in the operations that follow.

According to the design flow of the framework, the output image from this implementation is an input data to the next phrase in the framework (image sharpening stage). Therefore, these output images were passed to the next stage for image deblurring process.

4.1.2 Results for Image Deblurring

The section discusses and analyses the experimental results for removal of image blur that was achieved by recovering a sharp image from the corresponding blurred images.

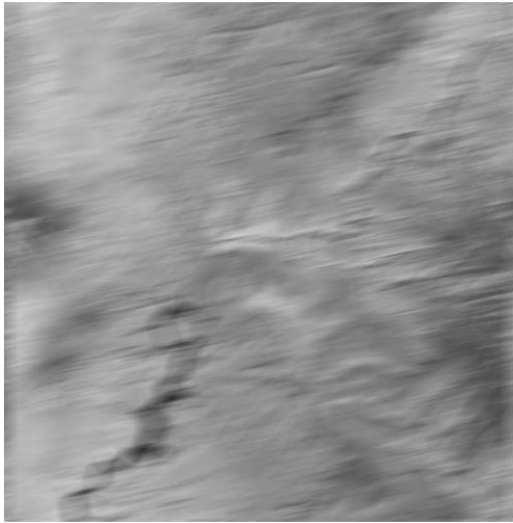


Figure 4. 5 1st date image (blurred)

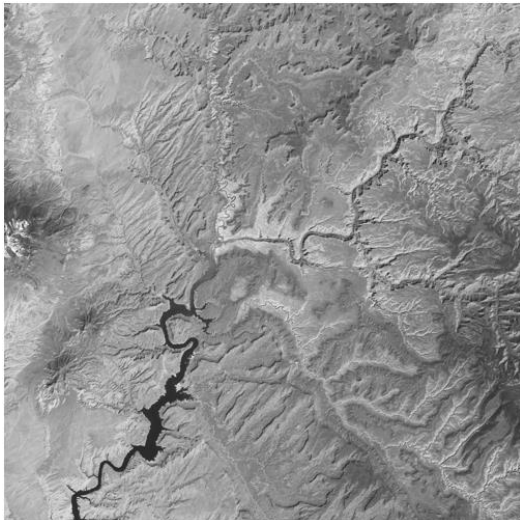


Figure 4. 6 1st date image (Restored image)

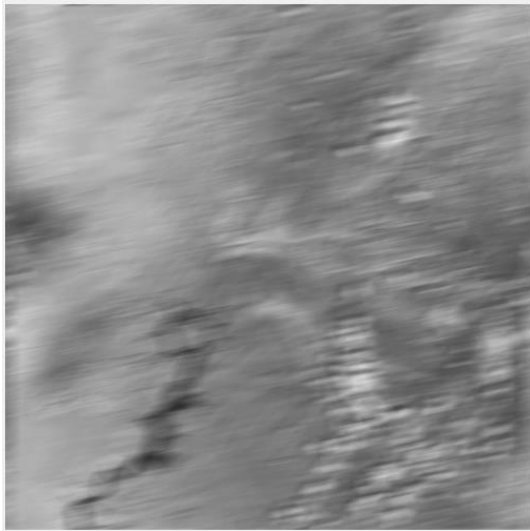


Figure 4. 7 2nd date image (blurred)

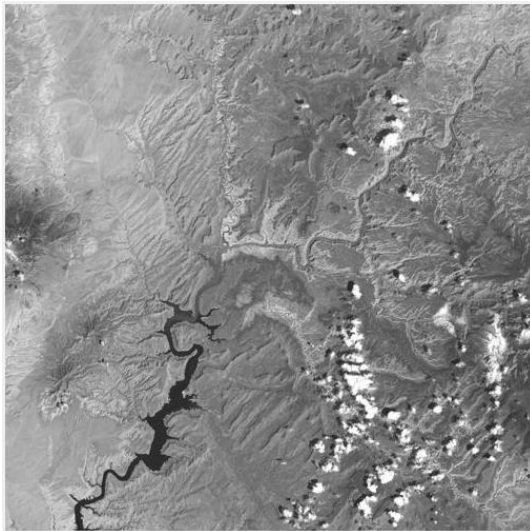


Figure 4. 8 2nd date image (Restored image)

Given the blurred images, Figure 4. 5 and Figure 4. 7 the goal of the experiment was to restore the degraded images by generating for each image an approximate as close to the original

image as possible. These blurred images show a lot of distortion, and the shape of objects or regions within the image are distorted to a state that is not recognizable. This state of distortion or blur render the images unusable for most applications especially computer vision. The deblurring operation has removed the blur, restoring the original images. The regions and objects in the images are now very clear and recognizable by visual interpretation. The lake, the snow and all other parts of the images stand out clearly. The output images are close estimates of the original images and this indicates that the deblurring operation was very effective in removing the blur.

The deblurring algorithm applied on the blurred images, produced the best estimate of the original images shown in Figure 4. 6 and Figure 4. 8 respectively. The results also show that the output images were noise free; there is no salt and pepper appearing on both images. This is an indication that the previous stage (denoising) has adequately removed the image noise though this result could not be observed due to the presence of image blur at that stage.

The experimental results of the deblurring process indicate that the deconvolution process in this framework was successful in producing the expected results. The satisfactory success of this experiment validates the deblurring algorithm performed in the Xilinx model-based design. The results in this experiment lead to the conclusion that the deblurring process is a necessary and required image restoration operation for removing the image blur. For this reason, this operation merits to be placed in the framework, and the omission of the deblurring process in the framework can lead to inaccurate results in the change detection framework due to blur impurities.

4.1.3 Results of Image Sharpening Process

This section discusses the experimental results obtained from the FPGA implementation of the image sharpening using XSG in Chapter 3. The aim of this implementation is to produce sharp images by performing blurring and masking techniques.

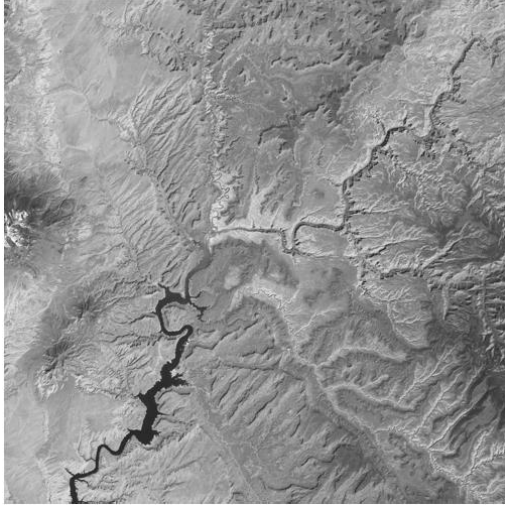


Figure 4. 9 1st date image

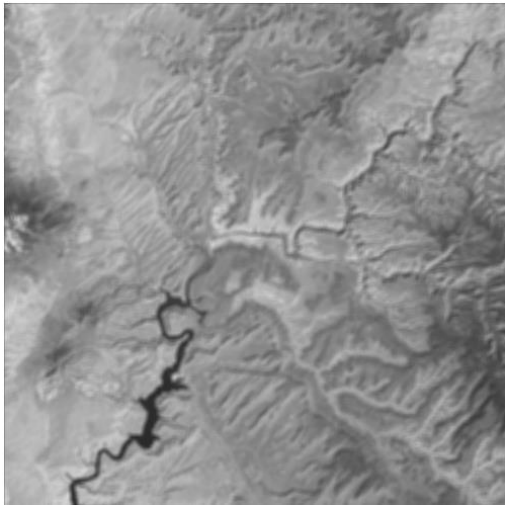


Figure 4. 10 1st date image (slightly blurred)



Figure 4. 11 Mask (1st date image)



Figure 4. 12 2nd date image



Figure 4. 13 2nd date image (slightly blurred)



Figure 4. 14 2nd date image (Mask)

In this image sharpening implementation, images Figure 4. 9 and Figure 4. 12 (representing outputs from the deblurring stage) were slightly blurred to produce images Figure 4. 10 and Figure 4. 13

respectively. These blurred images were then subtracted from the respective original images to obtain the mask (1st date image) Figure 4. 11 and mask (2nd date image) Figure 4. 14.

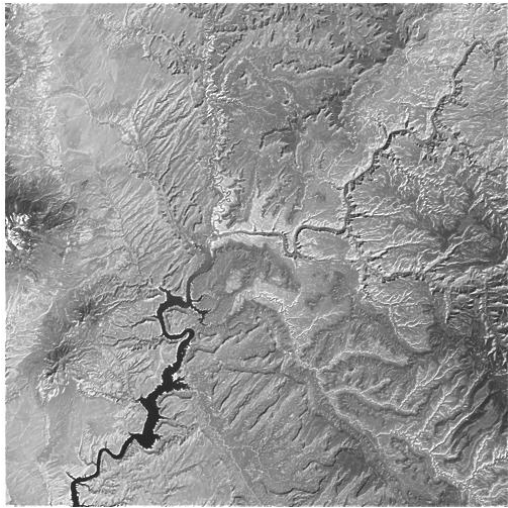


Figure 4. 15 Sharpened image

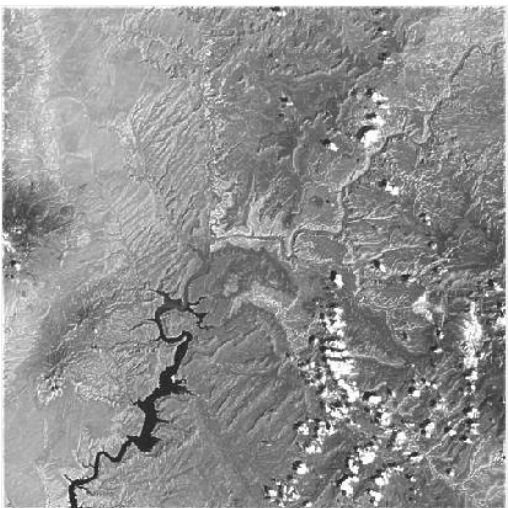


Figure 4. 16 Sharpened image

The mask images (difference images) were then added to the corresponding original images and generated sharpened images.

The results of the FPGA implementation indicate that the image sharpening process using Xilinx/Simulink model-based design was successful and produced satisfactory outputs, Figure 4. 15 and Figure 4. 16. The addition process of the mask image to the original image has produced a more sharpened image than the original, thus validating the image sharpening algorithm;

$$f_s(x, y) = f_o(x, y) + \Omega * f_d(x, y)$$

In this implementation the weighted mask value Ω , is 1 from the Xilinx constant block.

The output images or sharpened images showing the lake, streams, rocks, vegetation and some white patches are visually clear and they are of a superior quality. The areas or objects in the image stand out distinctively and can be easily identified. The sharpening operation made the details of the images to be well-defined in their extensive range of gray shades and white color. This high-quality output is good for visual interpretation and further image processing. The next stage in this framework is the edge detection, and this good quality image data made the extraction of lines, corners and edges in the images to be much easy for the edge detection algorithm.

4.1.4 Results for the Sobel Edge Detection

The principal purpose of the Sobel edge detection was to prepare input images for further processing in the change detection phase. The input data to this phase is the two sharpened images from the image sharpening stage of the framework. The experimental results from edge detection operation indicate that the implementation of the Sobel edge detection model-based design was successful in extracting edges in the two images.

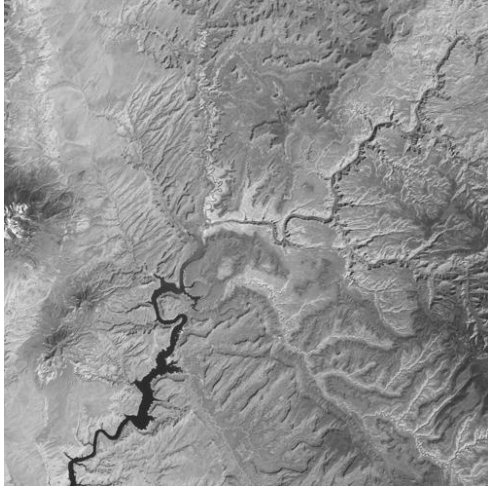


Figure 4. 17 1st date image

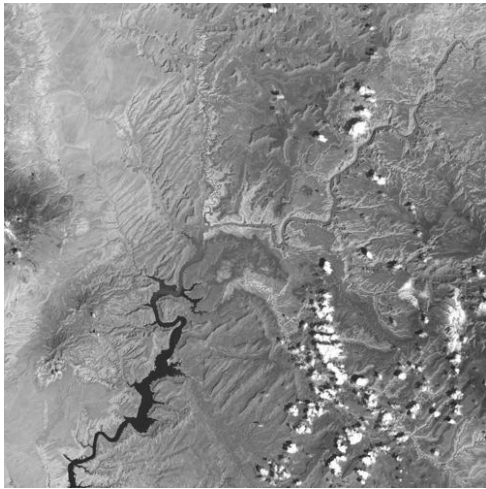


Figure 4. 18 2nd date image

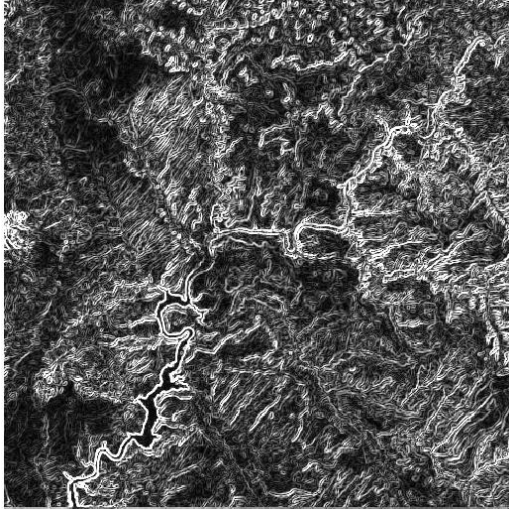


Figure 4. 19 1st date image (Output image with edges detected)

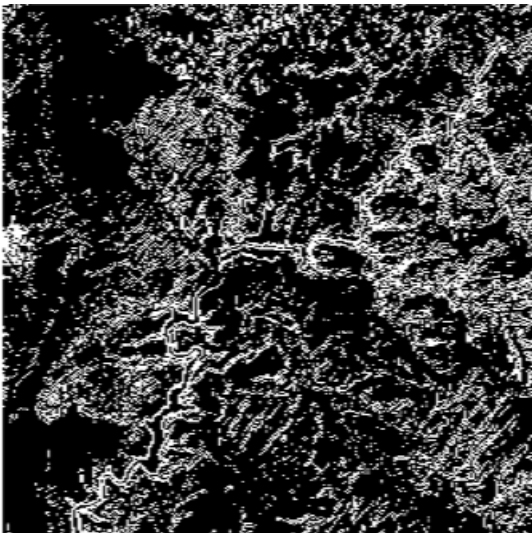


Figure 4. 20 Binary 1st date image

The edges were extracted from the input (1st date image) shown in Figure 4. 17. The output image is Figure 4. 19 and the extracted corners, edges and lines are highly noticeable. This output was then converted to binary format, Figure 4. 20. In this format, the image was now binary data and its contents were classified as black or white representing bit 1 for white and 0 for black, Figure 4. 20. This output indicated that the binary conversion was successful. The result was a verification that the input data to the next stage (image subtraction) was appropriate and suitable for change detection by an XOR operator.

The output image corresponding to 2nd date image is shown in Figure 4. 21 with its edges extracted. The edges, lines and corners were revealed, clearly showing the outline or bounds of the corresponding objects or areas in the input images. The edge detection operation has left no part of the image untouched, every visible object in the input images such as the contours, the small patches of snow and the streams have their outlines extracted. The lines are predominately white, and the background of the image has a variation of gray. The edge detection output, which was converted to binary data, Figure 4. 22 was successful, and its contents appeared as black or white representing bit 1 for white and 0 for black. The success of the binary conversion on the 2nd date image provided a confirmation that the input image data to the next stage was appropriate, meeting input requirements of the XOR operator.

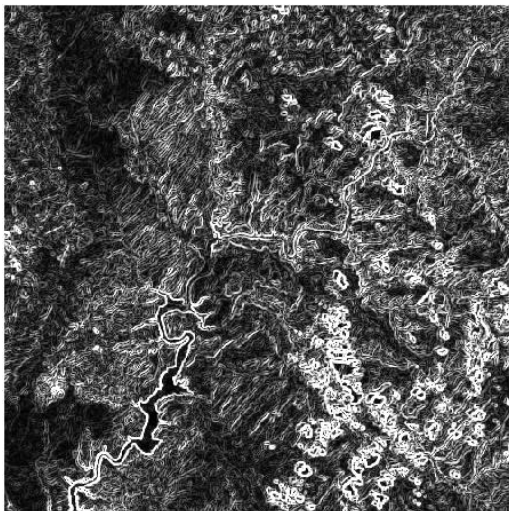


Figure 4. 21 2nd date image (Output image with edges detected)

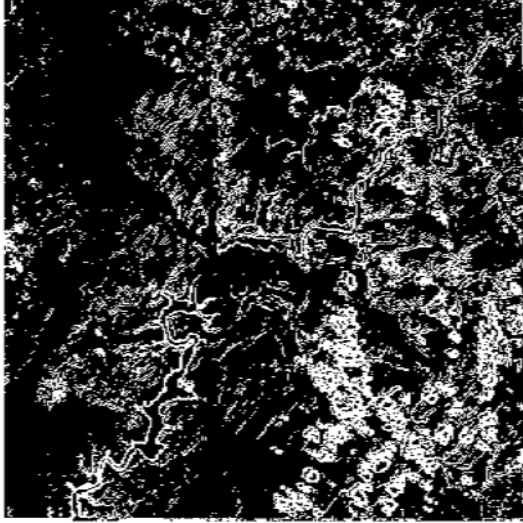


Figure 4. 22 Binary 2nd date image

The pronounced extracted edges, corners and lines in the output images indicate that the Sobel edge detection implementation was a success, thus validating the Sobel edge detection algorithm. When the input unit to the image subtraction block receives the images with such pronounced edges, corners and lines, then the quality of the out-image data from the subsequent operation is expected to be extraordinary.

4.2 Results for Image Subtraction Simulation Using the ISE

In Chapter 3.6.5 change detection was implemented using binary image subtraction. This process as explained in previous chapter is carried out by the exclusive-OR (XOR) operator. The XOR operator schematic for image subtraction was designed and simulated using the ISE.

The simulation output from the ISE is presented in the form of waveforms. The following screen shots show the output waveforms from the ISE simulator. The operator takes two input discrete signals with value pairs of 0 1, 0 0, 1 0 and 1 1 and the period set for this schematic is 10ns. The two inputs are designated at A and B, while the output is designated as C. The upper edge of the

waveform indicates the magnitude of value '1' and the lower edge of the waveform indicates the magnitude of value '0'.

There are four output waveforms for the simulation representing the four input signal pairs, and each graph will be described below.

4.2.1 Output Waveform for Input Signal pair '0 0'

The output waveform for the pair '0 0' is generated from input signal A and B, both have values of digit '0', and the output C has a value '0', Figure 4. 23. The discrete wave indicates low edges for both inputs and a high edge for the output.

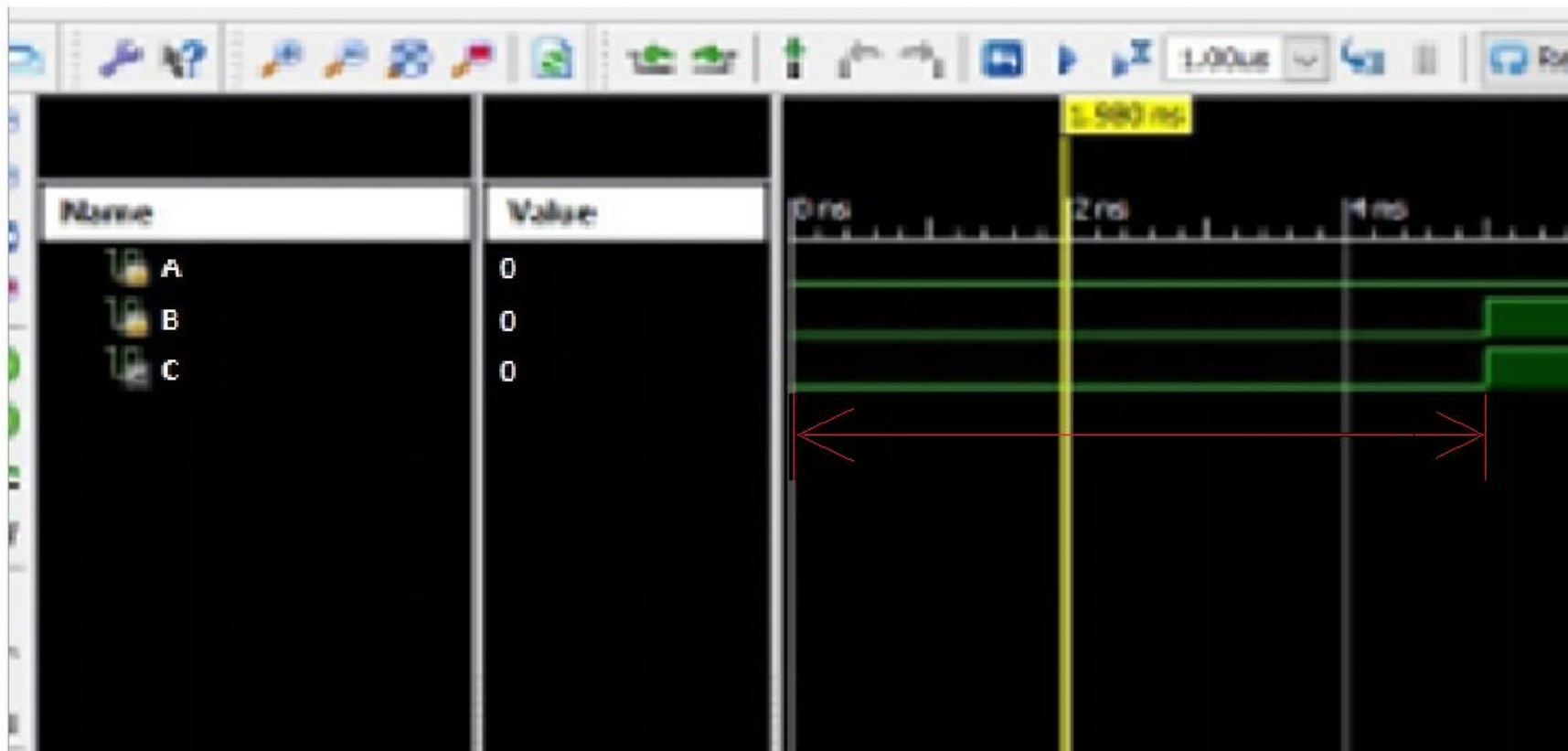


Figure 4. 23 Waveform highlighting the output and input signal pair '0 0'

4.2.2 Output Waveform for Input Signal pair '0 1'

The output waveform for the pair '01 ' is shown in Figure 4. 24, the signal A has a value of '0' and B has a value of digit '1', and the output C has a value '1'. The discrete wave indicates a low edge for input A, high edge for input B and a high edge for the output C.

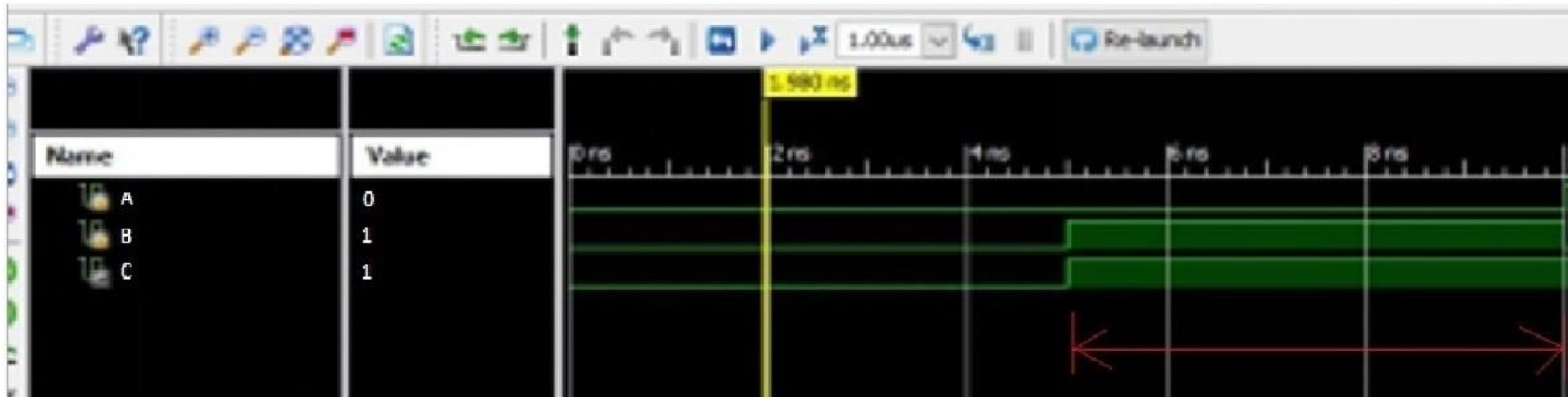


Figure 4. 24 Waveform highlighting the output and input signal pair '0 1'

4.2.3 Output Waveform for Input Signal pair '1 0'

The output waveform for the pair '1 0 ' is shown in Figure 4. 25, the signal A has a value of '1' and B has a value of digit '0', and the output C has a value '1'. The discrete wave indicates a high edge for input A, low edge for input B and a high edge for the output C.

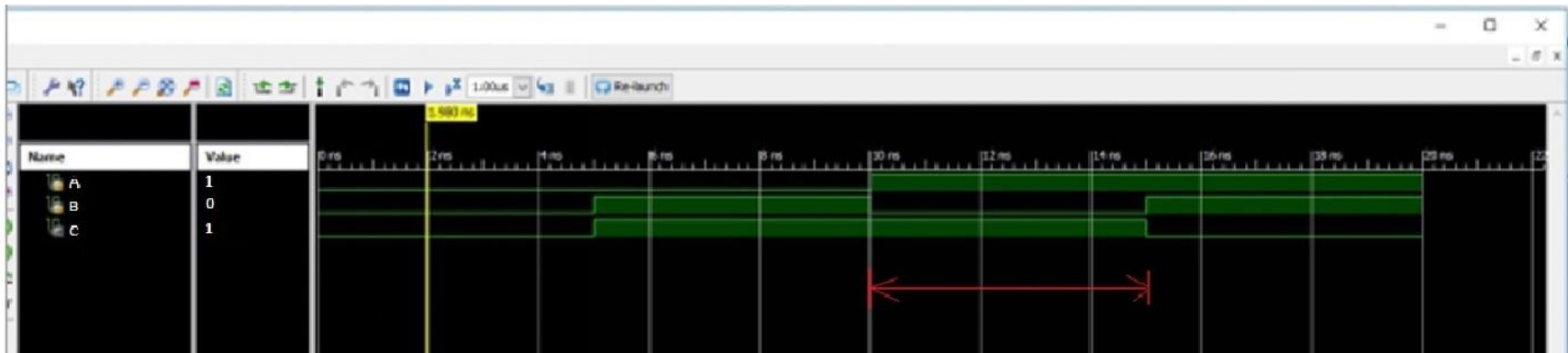


Figure 4. 25 Waveform highlighting the output and input signal pair '1 0'

4.2.4 Output Waveform for Input Signal pair '1 1'

The output waveform for the pair '1 1' is shown in Figure 4. 26, the signal A has a value of '1' and B has a value of digit '1', and the output C has a value '0'. The discrete wave indicates high edges for both inputs and a low edge for the output.

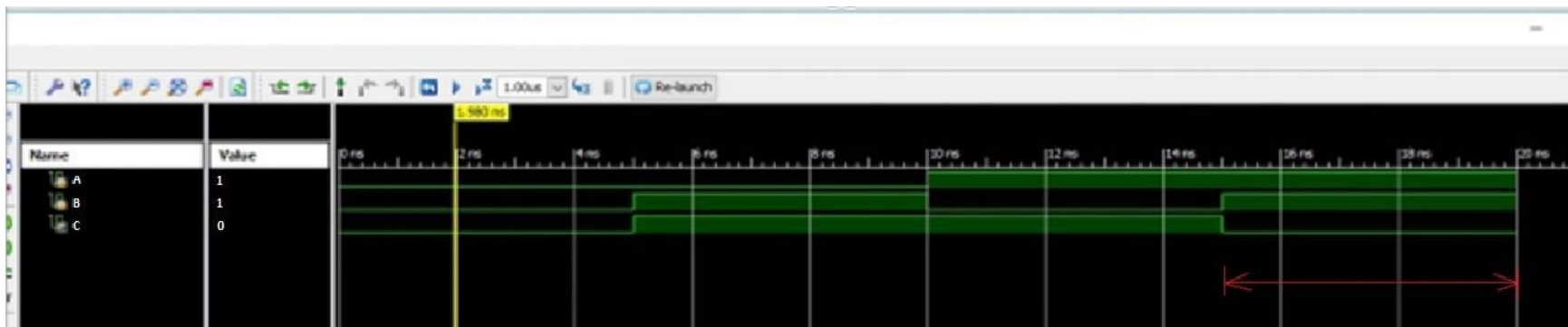


Figure 4. 26 Waveform highlighting the output and input signal pair '1 1'

The output generated in each of the waveforms coincides with values of the truth table for the bitwise XOR operator which returns 1 if operands are unequal, and returns 0 if both operands are the same. These outputs validate the output derived from the image subtraction XOR operation where the output is only high if input values are different, else the output is low.

The output waveforms from Figure 4. 23 to Figure 4. 26 indicate the properties of the output image when XOR operation is applied to the two binary images (white and black pixels). The binary 1 represents the white pixel and binary 0 represents the black pixels in both the input and out images. In this implementation if the pixels of the 1st date image and the pixels of the 2nd date image are complementary to each other, the output image pixel becomes black, otherwise the output image pixel becomes white. The black pixels in the output image indicate that the corresponding pixels on both the 1st date image and the 2nd date image did not change or remained the same. When comparing the newer image to the older image, the pixels which did not change in the new images produce a "0" and this corresponds to black pixels. For the pixels that have changed, a '1' is produce and this corresponds to white pixels.

4.3 Results of the Change Detection Framework

The experimental results, which were derived from the FPGA implementation of image change detection, conducted using the Simulink/XSG model-based design. The image shown in Figure 4. 27 is the output of the change detection framework that was generated through the image subtraction operation on the two binary images

This is the most important image data produced in all the operations that were executed in this framework. The framework's principal aim of detecting change between two images has been achieved through the image subtraction operation in this implementation.

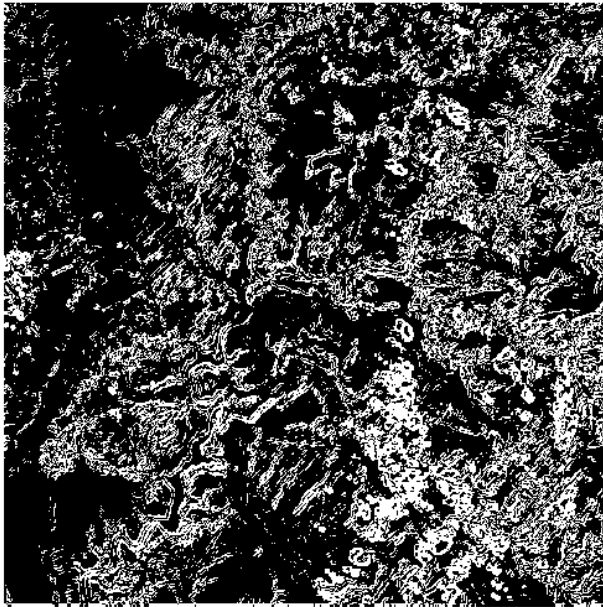


Figure 4. 27 Difference image (output of the change detection process)

The result shows that the algorithm has revealed or spotted the image differences. The differences between the two lake images which were obvious and visible to the naked eye were the white patches of snow in one of the original images, but the bitwise logical XOR operation in the image subtraction unit produces changes that are not as obvious to the naked eye as they are in binary format. The bitwise XOR operation preserves any corresponding image pixel pair if

the pixels have different values. This was used to highlight differences in 1st date binary image Figure 4. 20 and 2nd date image Figure 4. 22, respectively.

The image subtraction implementation uses the bitwise XOR logical operator which is a Boolean operator that works on binary data. The binary input images have only two colors (black and white). The bitwise XOR logical operator returns 1 if both bits are different, and returns 0 if both bits are the same, in this experiment considering the pixel value 255 (11111111 binary) representing white pixels and pixel value 0 (00000000 binary) representing black pixels. When the pixel values are the same, the out is (0) and when the pixel values are different, the output is (255), therefore when two pixels are different, the output is a white pixel, and else the output is a black pixel.

A careful observation and analysis of the properties of the difference image coming from this phase indicate that the bitwise operation had been successfully executed on the images. The pixel values of first date binary image were XORed with the corresponding pixel values from the second date binary image. In the logical XOR bitwise operation each and every pixel value that did not change, yielded a zero(0) and all are represented as black in the output image, while every pixel value that have changed yielded a one (1), and all are portrayed as white in the output image. The resultant produced by the black and white pixels form the output image, which represents the differences between the two images. This technique presents a way of detecting changes between images.

4.4 Results for Contrast Stretching

The principal purpose of the contrast stretching stage is to get enhanced quality of the output image such that its quality is suitable for human visual interpretation. The input image to the contrast stretching model-based design was the output image from the image subtraction stage. This difference image has a low contrast, and it was transformed into a high-contrast image Figure 4. 28. The contrast stretching operation produced an image with improved contrast. The quality of the binary difference image has tremendously improved; showing much pronounced features that reveal the fine details of the image, Figure 4. 28.

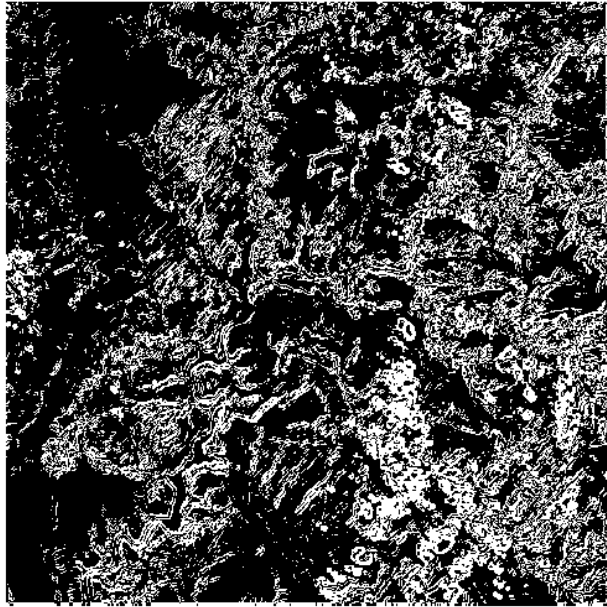


Figure 4. 28 Contrast Stretching

4.5 Framework Evaluation

The tests conducted in the design implementations were used to assess and determine the level of robustness of the framework conceptual model, and this was achieved through the widely adopted standard quality assessment metrics in image processing. An assessment occurs when the degree of fidelity of original image against the enhanced or restored image is measured. The degree of quality for input and out images was measured for the individual components of the framework. The quality performance parameters for justifying the results for this framework are the Peak Signal-to-Noise Ratio (PSNR) and the mean square error (MSE). The PSNR is a mathematical performance quality measure of the pixel difference between two images[98]. PSNR value is measured in decibels (dB). The typical values for the PSNR in lossy image are between 30 and 50dB. The larger the *PSNR* value the better the quality of the compressed, or reconstructed image[99].

The MSE is a performance quality measure which calculates the average value of the squares error or the deviation[100].

The MSE between the two signals is expressed as:

$$MSE = \frac{1}{N \times M} \sum_{i=0}^{N-1} \sum_{j=0}^{M-1} [X(i, j) - Y(i, j)]^2 \quad 3.31$$

Where X (i, j) is the original image and Y (i, j) is enhances or restored image

Let X and Y two arrays of size N x M.

i and j are the pixel position of the M×N image.

MSE is zero when $X(i, j) = Y(i, j)$.

$$PSNR = 10 \log_{10} \frac{L^2}{MSE} \quad 3.32$$

When MSE approaches zero or equals to zero, the similarity between the two images increases. The PSNR value tells how close the output image is to the reference image. For this framework, the MATLAB program was used to perform calculations of the PSNR and MSE values by applying equations 3.31 and 3.32 respectively, Listing 8. In the quality assessment of enhancement and restoration processes, gray scale images were used.

The findings from the framework implementations discussed in this chapter are very critical in addressing some of the research questions that guided the research work. The results also helped

to bring about proofs of some concepts discussed in the methodology of the thesis. The findings show that image enhancement and restoration techniques get rid of impurities, sharpen the image and extract edges making the new image easier to work with. When impurities such as noise and image blur are not removed from input images the quality of the results from any further image processing or analysis is reduced. It has also been observed that sequential order of the image pre-processing operations has a significant influence in the accuracy of the final output of the change detection process. In evaluating the framework components, it has been observed that the results from a change detection process applied on images that were not enhanced using denoising, sharpening and deblurring have diminished quality such as less pronounced edges. The overall quality of the difference image is inferior to the quality of the difference image resulting from a change detection process with enhanced input images. The overall quality assessments have shown that a change detection process produces higher quality results when image enhancement has been applied to inputs as observed in [17].

4.5.1 Image Denoising

For the image-denoising phase of the framework, the PSNR value observed for the denoised image/original image, and the original image/noised image comparison are shown in Table 4.1. The PSNR and MSE values for the original image/noised image comparison are 29.65 B and 213.3 respectively and they indicate that the quality of the image has gone down after the salt and pepper was added to the original image. The PSNR and MSE values for the original image/denoised image comparison are 30.16 and 189.51 respectively and they indicate that the quality of the input image was greatly improved. The MSE dropped from 213.3 to 189.51, indicating an improved quality. The PSNR increased by few decibels, showing an improvement. The PSNR and MSE values in Table 4.2 display the same indication that, the quality has improved. These values imply the enhancement has brought higher improvement in the quality of the image after the noise removal. This fact is an indication that the image denoising component of the framework has performed as expected. These results of the denoising process evaluation show that the process has successfully achieved its role in the framework

Table 4. 1 PSNR and MSE for Image Denoising (1st date image)

Noise Type	ORIGINAL IMAGE - noised		Original -DENOISED IMAGE	
	PSNR(dB)	MSE(dB)	PSNR(dB)	MSE(dB)
Salt & Pepper noise	29.65	213.3	30.16	189.51

Table 4. 2 PSNR and MSE for Image Denoising (2nd date image)

Noise Type	ORIGINAL IMAGE - noised		Original -DENOISED IMAGE	
	PSNR(dB)	MSE(dB)	PSNR(dB)	MSE(dB)
Salt & Pepper noise	29.37	227.49	29.53	219.15

4.5.2 Image Deblurring

A quantitative performance measure achieved by the deblurring algorithm model is achieved by computing the PSNR and the MSE values between the input image and blurred image, then the input image and deblurred image. The calculations were performed using MATLAB and are shown in Table 4.3 and 4.4.

Table 4. 3 PSNR and MSE for Image Deblurring (1st date image)

original image / blurred image		Original/deblurred image	
PSNR(dB)	MSE(dB)	PSNR(dB)	MSE(dB)
29.65	213.30	30.16	189.51

Table 4. 4 PSNR and MSE for Image Deblurring (2nd date image)

original image / blurred image		Original/deblurred image (2 nd image)	
PSNR(dB)	MSE(dB)	PSNR(dB)	MSE(dB)
29.65	213.30	30.16	189.51

In Table 4.3, for the original image / blurred image, PSNR is 29.65 and MSE 213.30. For the original image / deblurred image comparison PSNR is 29.65 and MSE 213.30. The value of PSNR is larger in the original image / deblurred image column for both the 1st date and 2nd date images. This indicates that quality has improved. The MSE value is lower in the original image / deblurred image column for both the 1st date and 2nd date images. This measure confirms the improvement of the image after image blur was removed. The results in Table 4.4 shows the same decrease in the MSE values and the same increase in PSNR values. The quality has improved after deblurring in both input images. These results are an indication that the image deblurring component of the framework has performed as expected.

4.5.3 Image Sharpening

Evaluation of the image-sharpening phase of the framework included the calculation of the peak signal-to-noise ratio and the mean square error using MATLAB, Listing 8. The PSNR and MSE values for the original image/sharpened image comparison are 28.69 and 265.56 respectively as shown in Table 4.5. The values of both the PSNR and MSE indicate that the quality of the input image was greatly improved by using the unsharp-masking filter. The image features were accentuated, and image details appeared much clearer, so the visual inspection confirms the quality improvement by the image sharpening phase of the framework.

Table 4. 5 PSNR and MSE for Image Sharpening

original image / unsharpened image		Original/sharpened image	
PSNR (dB)	MSE(dB)	PSNR (dB)	MSE(dB)
28.69	265.56	30.15	189.51

4.5.4 Image Edge Detection

In this phase of the framework, the measure of degree of quality of the input image against the output image was performed using both the PSNR and the MSE parameters. The PSNR and MSE values for the comparison between the original image and the image with detected edges are 32.03 and 123.30 respectively as shown in Table 4.6. The PSNR and MSE values for the comparison between the sharpened image and the image with detected edges are 32.86 and 101.76 respectively. The calculations show an increase in the PSNR vale and a decrease in the MSE value. In this case, the quality has improved when edges were detected in a sharpened image as compared to an image with no sharpening.

Table 4. 6 PSNR and MSE for Image Edge Detection 1st date image)

original image / edges_image		sharpened image/edges_image	
PSNR(dB)	MSE(dB)	PSNR(dB)	MSE(dB)
32.03	123.30	32.86	101.76

Table 4. 7 PSNR and MSE for Image Edge Detection (2nd date image)

original image / edges image		sharpened image/edges_image	
PSNR(dB)	MSE(dB)	PSNR(dB)	MSE(dB)
31.53	138.18	31.53	138.13

The visual evaluation confirms the improvement of the image quality due to edge feature extraction that has revealed pronounced lines, corners and edges of the output image. This result is an indication that the image edge detection component of the framework has performed as expected.

The framework evaluation which was performed using the PSNR and MSE quality measure experiments has produced the required performance. When the original image was compared with a restored or enhanced image, the PSNR and MSE values showed an improvement in quality. The framework components for enhancement and restoration processes of the framework have performed as expected, showing quality improvements in the input and out images. This is an indication that when impurities such as noise and image blur are removed from the input images the overall quality of the image improves and this leads to improved results in other image processes that follow. Therefore, the evaluation has shown that the framework component stages

have satisfied the expected standard quality assessment. The algorithms implementations and evaluation results form the proof that the proposed conceptual model of the framework can be adopted into a framework for image change detection.

Chapter 5: Conclusion

5.1 Conclusion

This thesis presented an FPGA-based image change detection framework, which is a systematic and clearly laid out procedure for implementing image change detection using the Xilinx system generator or FPGA system. The change detection framework implementation was carried out and produced experimental results that successfully validated all the image processing algorithms in the entire framework.

In order to demonstrate the framework feasibility, all the mandatory stages; denoising, deblurring, sharpening, edge detection, image subtraction and contrast stretching were implemented thus validating the framework. The recommended methodical order was followed as depicted in the design flow.

The implementation of the proposed image processing operations was successful, and the results greatly supported the hypothesis and helped to achieve the expected output of the framework. The first stage of the framework (image enhancement and restoration), produced output images that were then denoised, deblurred and sharpened in this order. This stage helped to prepare the images for further analysis in the stages that follow. These enhanced images were then passed to the edge detection stages where the edges, corners and lines of the images were extracted. The extraction of more edge details in the images by the Sobel edge detection operation increased the effectiveness of the image change detection process that performed a pixel-by-pixel (bitwise) operation to pick or spot the differences between the two images. The successful extraction of the fine details of the difference image indicate that the preceding image processing technique (Sobel edge detection) is a desired step and a necessary technique for preparing the input image to the change detection process that revealed the distinguishing features between

the two seemingly identical images. The entire results set from the pre-processing stage (image enhancement and restoration) to the spot the difference stage (image change detection) have fully achieved the objectives of the research study. The results that came from all the stages of the proposed framework are consistent with the expected output and they all validate the algorithms in the framework. Therefore, these satisfactory experimental results confirm the validity of the FPGA-based framework, which was implemented using the Xilinx system generator. The exclusive-OR operation simulation results have been validated using experimental results obtained through the Xilinx logical XOR block unit.

The output image is the difference between 1st date image and 2nd date image. Some image differences were not easy to spot with a naked eye. Therefore, further results interpretation and analysis could help to evaluate the performance of the framework, especially the image subtraction phase.

While the change detection framework implementation was successful in producing satisfactory results, there is a need for more experiments to explore different image filtering algorithms that could improve the results. Insignificant changes could pose a challenge to visual interpretation or analysis especially when very tiny changes had occurred in image data. A statistical form of data interpretation could do a better job in analyzing of complex results. The other technique that could be helpful in analyzing the image difference is the use of Machine Learning algorithms.

The FPGA based image change detection framework developed and implemented in this thesis has proved to be a success. There were challenges especially the selection of the appropriate enhancement and restoration operations and their systematic order in the framework. This framework has the potential to have a huge impact in the field especially the ways of performing image pre-processing techniques and the order of pre-processing operations.

5.2 Future Work

In this framework, each image-processing phase has a specific and unique task it performs, and some image processing techniques could be employed to further improve the quality of the input image especially in the enhancement and restoration stage. Image registration is one of the image processing operations that is necessary in the preprocessing stage, but it was not implemented

because the input images from the dataset that were selected for the implementation did not require image alignment or registration.

The image change detection framework is built from model-based block sets in the MATLAB/Simulink and the Xilinx design environment targeting the FPGA system. It would have been ideal if this framework could have also been implemented using VHDL only targeting the Xilinx FPGA, in that way more results could have produced and helped to evaluate different framework designs.

The image change framework was developed using image enhancement, edge detection and image subtraction algorithms for spotting images differences. In future work, the same design could be tailored for implementing image processing operations for finding image differences between video frames/images such as image segmentation and background subtraction. The study limitations exist due to the constraints on framework design and methodology. However, these factors did not affect the findings of the study but leave room for exploring other alternative techniques especially the image pre-processing processes. It is reasonable to conclude that the use of the four image pre-processing techniques is required in preparing the input images for an improved change detection.

The quality assessments were applied on images before and after the enhancement and restoration processes and then visual observation was used to affirm the evaluation. This process could not be applied to the difference image due to the nature of the image manipulation, hence no image comparison was applied. The output could only be inspected visually, and this brought about a limitation to the evaluation.

While the implementations described in this thesis only focused on few images, numerous types of digital images exist that may require additional enhancement processes prior to implementing an image subtraction process. The unanticipated challenges and constraints on methodology could not permit numerous implementations for various types of images. The findings in this thesis show that an FPGA-based change detection framework has been implemented successfully, but there is room for improvement to this work that would help better the framework and improve the results.

Reference

- [1] R. C. Gonzalez and R. E. Woods, *Digital Image Processing*, 2nd ed. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 1992.
- [2] A. Huertas and R. Nevatia, "Detecting changes in aerial views of man-made structures," in *Sixth International Conference on Computer Vision (IEEE Cat. No.98CH36271)*, Jan. 1998, pp. 73–80, doi: 10.1109/ICCV.1998.710703.
- [3] J. B. Collins and C. E. Woodcock, "An assessment of several linear change detection techniques for mapping forest mortality using multitemporal landsat TM data," *Remote Sens. Environ.*, vol. 56, no. 1, pp. 66–77, Apr. 1996, doi: 10.1016/0034-4257(95)00233-2.
- [4] R. T. Collins, A. J. Lipton, and T. Kanade, "Introduction to the special section on video surveillance," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 22, no. 8, pp. 745–746, Aug. 2000, doi: 10.1109/TPAMI.2000.868676.
- [5] S. Kunić and Z. Šego, "Beyond HDTV technology," in *Proceedings ELMAR-2013*, Sep. 2013, pp. 83–87.
- [6] L. Medina *et al.*, "A Comparison of FPGA and GPGPU Designs for Bayesian Occupancy Filters," *Sensors*, vol. 17, no. 11, Nov. 2017, doi: 10.3390/s17112599.
- [7] M. Gorgoń, "Parallel performance of the fine-grain pipeline FPGA image processing system," *Opto-Electron. Rev.*, vol. 20, no. 2, pp. 153–158, Jun. 2012, doi: 10.2478/s11772-012-0021-2.
- [8] O. Storaasli, W. Yu, D. Strenski, and J. Maltby, "Performance Evaluation of FPGA-Based Biological Applications," p. 15.
- [9] W. J. MacLean, "An Evaluation of the Suitability of FPGAs for Embedded Vision Systems," in *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05) - Workshops*, Sep. 2005, pp. 131–131, doi: 10.1109/CVPR.2005.408.
- [10] D. L. C. author, P. Mausel, E. Brondizio, and E. Moran, "Change detection techniques," *Int. J. Remote Sens.*, vol. 25, no. 12, pp. 2365–2401, Jun. 2004, doi: 10.1080/0143116031000139863.
- [11] H. A. Afify, "Evaluation of change detection techniques for monitoring land-cover changes: A case study in new Burg El-Arab area," *Alex. Eng. J.*, vol. 50, no. 2, pp. 187–195, Jun. 2011, doi: 10.1016/j.aej.2011.06.001.
- [12] K. Benkrid, D. Crookes, and A. Benkrid, "Towards a general framework for FPGA based image processing using hardware skeletons," *Parallel Comput.*, vol. 28, no. 7–8, pp. 1141–1154, Aug. 2002, doi: 10.1016/S0167-8191(02)00106-0.
- [13] Z. Salcic and J. Sivaswamy, "IMECO: A RECONFIGURABLEFPGA-BASEDIMAGE ENHANCEMENT CO-PROCESSOR FRAMEWORK," p. 4, 1997.

- [14] M. Rao, P. rajesh kumar, and M. A., "Image Processing using FPGAs: A framework," *Int. J. Comput. Technol. Appl.*, vol. 9, no. 19, pp. 9191–9197, Sep. 2016.
- [15] G. Singh and A. Mittal, "Various Image Enhancement Techniques - A Critical Review," *Int. J. Innov. Sci. Res.*, vol. 10, no. 2, pp. 267–274, Oct. 2014.
- [16] M. A. Covington, *Astrophotography for the amateur*, 2nd ed. Cambridge ; New York: Cambridge University Press, 1999.
- [17] Department of Electrical and Computer Engineering, Isfahan University of Technology, 8415683111 Isfahan, Iran, M. Reza Mahmoodi, and S. Masoud Sayedi, "A Low Cost High Speed FPGA-Based Image Processing Framework," *Int. J. Mod. Educ. Comput. Sci.*, vol. 8, no. 3, pp. 11–21, Mar. 2016, doi: 10.5815/ijmecs.2016.03.02.
- [18] A. Nieto, V. M., and D. L., "An FPGA-based Topographic Computer for Binary Image Processing," in *Image Processing*, Y.-S. Chen, Ed. InTech, 2009.
- [19] A. P. D. Binotto, D. Doering, T. Stetzelberger, P. McVittie, S. Zimmermann, and C. E. Pereira, "A CPU, GPU, FPGA System for X-Ray Image Processing Using High-Speed Scientific Cameras," in *2013 25th International Symposium on Computer Architecture and High Performance Computing*, Oct. 2013, pp. 113–119, doi: 10.1109/SBAC-PAD.2013.1.
- [20] R. Cieszewski, K. Pozniak, and R. Romaniuk, "Review of parallel computing methods and tools for FPGA," *Proc SPIE Vol8903*, vol. 8903, pp. 8903–62, Jan. 2013.
- [21] F. Bhatti, T. Greiner, M. Heizmann, and M. Ziebarth, "A new FPGA based architecture to improve performance of deflectometry image processing algorithm," in *2017 40th International Conference on Telecommunications and Signal Processing (TSP)*, Jul. 2017, pp. 559–562, doi: 10.1109/TSP.2017.8076049.
- [22] Meng Li *et al.*, "High efficiency VLSI implementation of an edge-directed video up-scaler using high level synthesis," in *2015 IEEE International Conference on Consumer Electronics (ICCE)*, Jan. 2015, pp. 92–95, doi: 10.1109/ICCE.2015.7066333.
- [23] W. Sun, M. J. Wirthlin, and S. Neuendorffer, "FPGA Pipeline Synthesis Design Exploration Using Module Selection and Resource Sharing," *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, vol. 26, no. 2, pp. 254–265, Feb. 2007, doi: 10.1109/TCAD.2006.887923.
- [24] C. Grozea, Z. Bankovic, and P. Laskov, "FPGA vs. Multi-core CPUs vs. GPUs: Hands-On Experience with a Sorting Application," in *Facing the Multicore-Challenge*, vol. 6310, R. Keller, D. Kramer, and J.-P. Weiss, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 105–117.
- [25] S. Mittal, S. Gupta, and S. Dasgupta, "FPGA: An Efficient And Promising Platform For Real-Time Image Processing Applications," Feb. 2008.
- [26] B. Zeidman, *Designing with FPGAs and CPLDs*. Lawrence, Kan: CMP, 2002.
- [27] S. D. Brown, "An overview of technology, architecture and CAD tools for programmable logic devices," in *Proceedings of IEEE Custom Integrated Circuits Conference - CICC '94*, May 1994, pp. 69–76, doi: 10.1109/CICC.1994.379764.

- [28] N. Tara, L. Jamal, and H. M. H. Babu, "An efficient approach to design a compact reversible programmable logic array," in 2015 IEEE International WIE Conference on Electrical and Computer Engineering (WIECON-ECE), Dec. 2015, pp. 135–138, doi: 10.1109/WIECON-ECE.2015.7444017.
- [29] M. Sonka, V. Hlavac, and R. Boyle, Image processing, analysis and machine vision (3. ed.). 2008.
- [30] T. Celik, "Change Detection in Satellite Images Using a Genetic Algorithm Approach," IEEE Geosci. Remote Sens. Lett., vol. 7, no. 2, pp. 386–390, Apr. 2010, doi: 10.1109/LGRS.2009.2037024.
- [31] A. Boyat and B. K. Joshi, "Image denoising using wavelet transform and median filtering," in 2013 Nirma University International Conference on Engineering (NUICONE), Nov. 2013, pp. 1–6, doi: 10.1109/NUICONE.2013.6780128.
- [32] M. Sonka, V. Hlavac, and R. Boyle, "Image pre-processing," in Image Processing, Analysis and Machine Vision, M. Sonka, V. Hlavac, and R. Boyle, Eds. Boston, MA: Springer US, 1993, pp. 56–111.
- [33] R. L. Lagendijk and J. Biemond, "Chapter 14 - Basic Methods for Image Restoration and Identification," in The Essential Guide to Image Processing, A. Bovik, Ed. Boston: Academic Press, 2009, pp. 323–348.
- [34] P. Sharma, G. Singh, and A. Kaur, "Different Techniques Of Edge Detection In Digital Image Processing," Int. J. Eng., vol. 3, no. 3, p. 4, 2013.
- [35] M. Boo, E. Antelo, and J. D. Bruguera, "VLSI implementation of an edge detector based on Sobel operator," in System Architecture and Integration Proceedings of Twentieth Euromicro Conference, Sep. 1994, pp. 506–512, doi: 10.1109/EURMIC.1994.390364.
- [36] Y. Said, T. Saidani, F. Smach, and M. Atri, "Real time hardware co-simulation of Edge Detection for video processing system," in 2012 16th IEEE Mediterranean Electrotechnical Conference, Mar. 2012, pp. 852–855, doi: 10.1109/MELCON.2012.6196563.
- [37] Y. Said, T. Saidani, F. Smach, and M. Atri, "Real time hardware co-simulation of Edge Detection for video processing system," in 2012 16th IEEE Mediterranean Electrotechnical Conference, Mar. 2012, pp. 852–855, doi: 10.1109/MELCON.2012.6196563.
- [38] N. Basheer, A. M Aaref, and D. Ayyed, "Digital Image Sobel Edge Detection Using FPGA," Jul. 2015.
- [39] B. You, W. Sheng, H. Ma, Y. Gu, and Y. Qin, "Implementation of Sobel Edge Detection on FPGA based on OpenCL," in 2017 IEEE 7th Annual International Conference on CYBER Technology in Automation, Control, and Intelligent Systems (CYBER), Jul. 2017, pp. 753–758, doi: 10.1109/CYBER.2017.8446103.
- [40] G. Anusha, D. T. J. Prasad, and D. D. S. Narayana, "Implementation of SOBEL Edge Detection on FPGA," Int. J. Comput. Trends Technol., p. 4, 2012.
- [41] "Mohapatra et al. - 2015 - Optimized approach of sobel edge detection techniq.pdf." .

- [42] S. R. Dixit and D. A. Y. Deshmukh, "Sobel Edge Detection Implementation using Spartan 3 FPGA and Xilinx System Generator," vol. 3, no. 4, p. 7, 2015.
- [43] V. Sanduja and R. Patial, "Sobel Edge Detection using Parallel Architecture based on FPGA," vol. 3, Jan. 2012.
- [44] A. G. Mahalle and A. M. Shah, "An Efficient Design for Canny Edge Detection Algorithm Using Xilinx System Generator," in 2018 International Conference on Research in Intelligent and Computing in Engineering (RICE), Aug. 2018, pp. 1–4, doi: 10.1109/RICE.2018.8509063.
- [45] C. Gentsos, C. Sotiropoulou, S. Nikolaidis, and N. Vassiliadis, "Real-time canny edge detection parallel implementation for FPGAs," in 2010 17th IEEE International Conference on Electronics, Circuits and Systems, Dec. 2010, pp. 499–502, doi: 10.1109/ICECS.2010.5724558.
- [46] S. Martins and J. C. Alves, "A high-level tool for the design of custom image processing systems," in 8th Euromicro Conference on Digital System Design (DSD'05), Aug. 2005, pp. 346–349, doi: 10.1109/DSD.2005.7.
- [47] P. Nguyen, J. Cho, and S. B. Cho, "An architecture for real-time hardware co-simulation of edge detection in image processing using Prewitt edge operator," in 2014 International Conference on Electronics, Information and Communications (ICEIC), Jan. 2014, pp. 1–2, doi: 10.1109/ELINFOCOM.2014.6914387.
- [48] Y. Li, Q. Yao, B. Tian, and W. Xu, "Fast double-parallel image processing based on FPGA," in Proceedings of 2011 IEEE International Conference on Vehicular Electronics and Safety, Jul. 2011, pp. 97–102, doi: 10.1109/ICVES.2011.5983754.
- [49] R. Harinarayan, R. Pannerselvam, M. M. Ali, and D. K. Tripathi, "Feature extraction of Digital Aerial Images by FPGA based implementation of edge detection algorithms," in 2011 International Conference on Emerging Trends in Electrical and Computer Technology, Mar. 2011, pp. 631–635, doi: 10.1109/ICETECT.2011.5760194.
- [50] D. Selvathi and J. Dharani, "Realization of Beamlet Transform edge detection algorithm using FPGA," in Image Processing Pattern Recognition 2013 International Conference on Signal Processing, Feb. 2013, pp. 131–135, doi: 10.1109/ICSIPR.2013.6497975.
- [51] ZhangShanshan and WangXiaohong, "Vehicle image edge detection algorithm hardware implementation on FPGA," in 2010 International Conference on Computer Application and System Modeling (ICCASM 2010), Oct. 2010, vol. 14, pp. V14-184-V14-188, doi: 10.1109/ICCASM.2010.5622351.
- [52] S. G. Kavitar, "FPGA based Image Feature Extraction Using Xilinx System Generator," vol. 5, p. 5, 2014.
- [53] A. J. Lingg, E. Zelnio, F. Garber, and B. D. Rigling, "A Sequential Framework for Image Change Detection," IEEE Trans. Image Process., vol. 23, no. 5, pp. 2405–2413, May 2014, doi: 10.1109/TIP.2014.2309432.

- [54] L. Bruzzone and F. Bovolo, "A Novel Framework for the Design of Change-Detection Systems for Very-High-Resolution Remote Sensing Images," *Proc. IEEE*, vol. 101, no. 3, pp. 609–630, Mar. 2013, doi: 10.1109/JPROC.2012.2197169.
- [55] A. J. Lingg, E. Zelnio, F. Garber, and B. D. Rigling, "A Sequential Framework for Image Change Detection," *IEEE Trans. Image Process.*, vol. 23, no. 5, pp. 2405–2413, May 2014, doi: 10.1109/TIP.2014.2309432.
- [56] A. Manan, "Implementation of image processing algorithm on FPGA," *Akgec J. Technol.*, vol. 2, no. 1, pp. 25–28, 2006.
- [57] V. K. Mishra, S. Kumar, and N. Shukla, "Image Acquisition and Techniques to Perform Image Acquisition," *SAMRIDDHI J. Phys. Sci. Eng. Technol.*, vol. 9, no. 01, Jul. 2017, doi: 10.18090/samriddhi.v9i01.8333.
- [58] S. Rahman, M. M. Rahman, K. Hussain, S. M. Khaled, and M. Shoyaib, "Image enhancement in spatial domain: A comprehensive study," in *2014 17th International Conference on Computer and Information Technology (ICCIT)*, Dec. 2014, pp. 368–373, doi: 10.1109/ICCITech.2014.7073123.
- [59] I. M. Hayder, H. A. Younis, and H. A.-K. Younis, "Digital Image Enhancement Gray Scale Images In Frequency Domain," *J. Phys. Conf. Ser.*, vol. 1279, p. 012072, Jul. 2019, doi: 10.1088/1742-6596/1279/1/012072.
- [60] B. Nguchu, "Critical Analysis of Image Enhancement Techniques," *Int. J. Electr. Electron. Res.*, vol. Vol. 4, p. pp: (23-33, Oct. 2016.
- [61] O. D. Mavrantza and D. Argialas, "Implementation and evaluation of spatial filtering and edge detection techniques for lineament mapping - Case study: Alevrada, Central Greece," *Proc. SPIE - Int. Soc. Opt. Eng.*, Mar. 2003, doi: 10.1117/12.463275.
- [62] D. Griffith, "Spatial Filtering," in *Handbook of Applied Spatial Analysis*, 2010, pp. 301–318.
- [63] A. Buades, B. Coll, and J.-M. Morel, "Image Denoising Methods. A New Nonlocal Principle," *SIAM Rev.*, vol. 52, pp. 113–147, Mar. 2010, doi: 10.1137/090773908.
- [64] S. S. Al-amri, N. V. Kalyankar, and S. D. Khamitkar, "A Comparative Study of Removal Noise from Remote Sensing Image," *ArXiv10021148 Cs*, Feb. 2010, Accessed: Sep. 09, 2019. [Online]. Available: <http://arxiv.org/abs/1002.1148>.
- [65] K. Khorsheed and O. Abood, "Survey of Denoising Techniques in Image Processing," vol. 2, pp. 18–23, Dec. 2018, doi: 10.5874/IJAMSR..2018.12.31.OGA.
- [66] S. K. Nayar and M. Ben-Ezra, "Motion-based motion deblurring," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 26, no. 6, pp. 689–698, Jun. 2004, doi: 10.1109/TPAMI.2004.1.
- [67] Huei-Yung Lin and Kun-Jhih Li, "Motion blur removal and its application to vehicle speed detection," in *2004 International Conference on Image Processing*, 2004. *ICIP '04.*, Oct. 2004, vol. 5, pp. 3407-3410 Vol. 5, doi: 10.1109/ICIP.2004.1421846.
- [68] P. Ganesan, G. Sajiv, and L. M. Leo, "Deblurring techniques for natural images — A comparative and comprehensive study," in *2017 Third International Conference on*

- Science Technology Engineering Management (ICONSTEM), Mar. 2017, pp. 873–878, doi: 10.1109/ICONSTEM.2017.8261363.
- [69] A. Mahalakshmi and B. Shanthini, “A survey on image deblurring,” in 2016 International Conference on Computer Communication and Informatics (ICCCI), Jan. 2016, pp. 1–5, doi: 10.1109/ICCCI.2016.7479956.
 - [70] A. Jain, D. Pandey, and S. Kumar, “PURPOSE OF BLIND DE-CONVOLUTION ALGORITHM FOR IMAGE RESTORATION,” Sep. 2019.
 - [71] A. Makandar and B. Halalli, “Image Enhancement Techniques using Highpass and Lowpass Filters,” *Int. J. Comput. Appl.*, vol. 109, no. 14, pp. 21–27, Jan. 2015, doi: 10.5120/19256-0999.
 - [72] “Digital Logic Introduction Using FPGAs | Elsevier Enhanced Reader.” <https://reader.elsevier.com/reader/sd/pii/S1877042815016456?token=188926BE513CD716E55BB7EEA68F621CC57DC4C641FFB9B0C7F07861F9578A660172709A8BD64BA9E02A51B7B1B74D29> (accessed Sep. 09, 2019).
 - [73] S. Dewangan and A. Sharma, “Image Smoothing and Sharpening using Frequency Domain Filtering Technique,” *Int. J. Emerg. Technol. Eng. Res.*, vol. 5, pp. 169–174, Apr. 2017.
 - [74] G. (Gianni) Ramponi, N. Strobil, S. K. Mitra, and T.-H. Yu, “Nonlinear unsharp masking methods for image contrast enhancement,” *J Electron. Imaging*, vol. 5, pp. 353–366, Jul. 1996, doi: 10.1117/12.242618.
 - [75] G. A. Yelpale and K. R. Desai, “IMPLEMENTATION OF GENERALIZED UNSHARP MASKING ALGORITHM FOR DIGITAL IMAGE,” p. 7, 2013.
 - [76] K. Sharma and A. Goyal, “Classification based survey of image registration methods,” in 2013 Fourth International Conference on Computing, Communications and Networking Technologies (ICCCNT), Jul. 2013, pp. 1–7, doi: 10.1109/ICCCNT.2013.6726741.
 - [77] M. V. Wyawahare, P. M. Patil, and H. K. Abhyankar, “Image Registration Techniques: An overview,” *Int. J. Signal Process. Image Process. Pattern Recognit.*, vol. 2, Sep. 2009.
 - [78] B. Zitová and J. Flusser, “Image registration methods: a survey,” *Image Vis. Comput.*, vol. 21, no. 11, pp. 977–1000, Oct. 2003, doi: 10.1016/S0262-8856(03)00137-9.
 - [79] Rashmi, M. Kumar, and R. Saxena, “Algorithm and Technique on Various Edge Detection : A Survey,” *Signal Image Process. Int. J.*, vol. 4, no. 3, pp. 65–75, Jun. 2013, doi: 10.5121/sipij.2013.4306.
 - [80] P. Selvakumar and S. Hariganesh, “The performance analysis of edge detection algorithms for image processing,” in 2016 International Conference on Computing Technologies and Intelligent Data Engineering (ICCTIDE’16), Jan. 2016, pp. 1–5, doi: 10.1109/ICCTIDE.2016.7725371.

- [81] G. Chaple and R. D. Daruwala, "Design of Sobel operator based image edge detection algorithm on FPGA," in 2014 International Conference on Communication and Signal Processing, Apr. 2014, pp. 788–792, doi: 10.1109/ICCSP.2014.6949951.
- [82] G. Chaple and R. D. Daruwala, "Design of Sobel operator based image edge detection algorithm on FPGA," in 2014 International Conference on Communication and Signal Processing, Apr. 2014, pp. 788–792, doi: 10.1109/ICCSP.2014.6949951.
- [83] "Albert Einstein by Yousuf Karsh | Huxley-Parlour Gallery." <https://huxleyparlour.com/albert-einstein-by-yousuf-karsh/> (accessed Sep. 19, 2019).
- [84] S. Minu and A. Shetty, "A Comparative Study of Image Change Detection Algorithms in MATLAB," *Aquat. Procedia*, vol. 4, pp. 1366–1373, 2015, doi: 10.1016/j.aqpro.2015.02.177.
- [85] S. K. Sarkar, A. K. De, and S. Sarkar, *Foundation of digital electronics and logic design*. Singapore: Pan Stanford Publications, 2015.
- [86] K. Wick, Morgan & Claypool Publishers, and Institute of Physics (Grande-Bretagne), *Applied digital logic exercises using FPGAs*. 2017.
- [87] R. Sultana and J. Samanta, "Comparison of different design techniques of XOR & AND gate using EDA simulation tool," *Int. J. VLSI Embed. Syst.*, vol. 04, May 2013.
- [88] B. Kaur, G. Kaur, and A. Kaur, "Detection and classification of Printed circuit board defects using image subtraction method," in 2014 Recent Advances in Engineering and Computational Sciences (RAECS), Mar. 2014, pp. 1–5, doi: 10.1109/RAECS.2014.6799537.
- [89] I. Petrescu, I.-B. Păvăloiu, and G. Drăgoi, "Digital Logic Introduction Using FPGAs," *Procedia - Soc. Behav. Sci.*, vol. 180, pp. 1507–1513, May 2015, doi: 10.1016/j.sbspro.2015.02.299.
- [90] K. B. Singh, T. V. Mahendra, R. S. Kurmvanshi, and C. V. R. Rao, "Image enhancement with the application of local and global enhancement methods for dark images," in 2017 International Conference on Innovations in Electronics, Signal Processing and Communication (IESC), Apr. 2017, pp. 199–202, doi: 10.1109/IESPC.2017.8071892.
- [91] S. S. Negi and Y. S. Bhandari, "A hybrid approach to Image Enhancement using Contrast Stretching on Image Sharpening and the analysis of various cases arising using histogram," in International Conference on Recent Advances and Innovations in Engineering (ICRAIE-2014), May 2014, pp. 1–6, doi: 10.1109/ICRAIE.2014.6909232.
- [92] R. Choudhary and S. Gawade, "Survey on Image Contrast Enhancement Techniques," vol. 2, no. 3, p. 5, 2016.
- [93] C. L. Chandrakar, M. Sharma, and S. K. Singhai, "SURVEY OF IMAGE CONTRAST ENHANCEMENT METHODS," 2012.
- [94] Murinto, S. Winiarti, D. P. Ismi, and A. Prahara, "Image enhancement using piecewise linear contrast stretch methods based on unsharp masking algorithms for leather image

- processing,” in 2017 3rd International Conference on Science in Information Technology (ICSITech), Oct. 2017, pp. 669–673, doi: 10.1109/ICSITech.2017.8257197.
- [95] “image processing – BlockoS.” <https://blog.blockos.org/?tag=image-processing> (accessed Sep. 18, 2019).
- [96] “lakepowell_20020512_lrg.jpg https://earthobservatory.nasa.gov/ContentWOC/images/lakepowell/lakepowell_20020512_lrg.jpg (accessed Sep. 20, 2019).
- [97] G. Liu et al., “Application of Near-Infrared Hyperspectral Imaging for Detection of External Insect Infestations on Jujube Fruit,” *Int. J. Food Prop.*, vol. 19, no. 1, pp. 41–52, Jan. 2016, doi: 10.1080/10942912.2014.923439.
- [98] Y. A. Y. Al-Najjar, “Comparison of Image Quality Assessment: PSNR, HVS, SSIM, UIQI,” vol. 3, no. 3, p. 6.
- [99] A. Baviskar, S. Ashtekar, and A. Chintawar, “Performance evaluation of high quality image compression techniques,” in 2014 International Conference on Advances in Computing, Communications and Informatics (ICACCI), Delhi, India, Sep. 2014, pp. 1986–1990, doi: 10.1109/ICACCI.2014.6968643.
- [100] Department of Electronics & Communication Engineering National Institute of Technical Teachers Training & Research, Chandigarh and R. Mehra, “Estimation of the Image Quality under Different Distortions,” *Int. J. Eng. Comput. Sci.*, Jul. 2016, doi: 10.18535/ijecs/v5i7.20.

APPENDIX: MATLAB M-Files and VHDL Codes

Listing 1: VHDL code for XOR operation for image subtraction simulation in
Xilinx ISE

```
--import std_logic from the IEEE library
```

```

library ieee;

use ieee.std_logic_1164.all;


-- ENTITY DECLARATION: name of entity, inputs, outputs, the gate takes 2
inputs

entity xor2 is

    port( A, B : in std_logic;

          C : out std_logic);

end xor2;


--FUNCTIONAL DESCRIPTION:

architecture func of xor2 is

begin

    C <= A xor B;

end func;


VHDL  testbench for the Exclusive-OR operation.


--import std_logic from the IEEE library

library ieee;

use ieee.std_logic_1164.all;


--ENTITY DECLARATION: no inputs, no outputs

entity xor2_tb is

end xor2_tb;


-- Testing the Exclusive-OR gate

architecture tb of xor2_tb is

```

```

--Passing xor2 entity to the testbench as component
component xor2 is
port( A, B : in std_logic;
      C : out std_logic);
end component;

signal inA, inB, outC : std_logic;
begin

--Mapping the testbench signals to the ports
mapping: xor2 port map(inA, inB, outC);

process
    --Tracking errors
    variable errCnt : integer := 0;
begin
    -- Test 1
    inA <= '0';
    inB <= '0';
    wait for 15 ns;
    assert(outC = '0') report "Error 1" severity error;
    if(outC /= '0') then
        errCnt := errCnt + 1;
    end if;

    -- Test 2
    inA <= '0';
    inB <= '1';
    wait for 15 ns;

```

```

    assert(outC = '1')    report "Error 2"    severity error;

    if(outC /= '1') then
        errCnt := errCnt + 1;
    end if;

--Test 3

inA <= '1';
inB <= '1';

wait for 15 ns;

assert(outC = '0')    report "Error 3"    severity error;

if(outC /= '0') then
    errCnt := errCnt + 1;
end if;

----- SUMMARY -----

if(errCnt = 0) then
    assert false report "Good!" severity note;
else
    assert true report "Error!"          severity error;
end if;

end process;

end tb;

-----

configuration cfg_tb of xor2_tb is

    for tb

        end for;

end cfg_tb;

```

-----END-----

Listing 2: Median filtering mcode (median.m)

```
i=imread('1stdate.jpg'); % 2nddate.jpg image is substituted here

if(size(i,3)~=1)

i=rgb2gray(I);

end

rr=0.1;

h=imnoise(i,'salt & pepper',rr);

[M,N]=size(h);

restored=h-h;

isp('*****Note:zero-padding*****');

disp('');

kernel_size=input('5');

k=zeros(kernel_size);

% Median filtering

start=floor(kernel_size*0.5);

for x=start:1:M-floor(kernel_size*0.5)

for y=start:1:N-floor(kernel_size*0.5)

% Coordinates in the kernel

x1=x-floor(kernel_size*0.5);

y1=y-floor(kernel_size*0.5);

% image pixels to the kernel

for p=1:1:kernel_size

for q=1:1:kernel_size

k(p,q)=h(x1+p-1,y1+q-1);

end

end
```

```

end

d=reshape(k,1,[]);

[r,c]=size(d);

for j=1:1:c-1
for i=1:1:c-1
    a=d(1,i);
    b=d(1,i+1);

    if(a>b)

        d(1,i)=b;

        d(1,i+1)=a;
    end
end
end

end

Median=d(1,floor(kernel_size*kernel_size*0.5)+1);

restored(x,y)=Median;

end

end

```

Listing 3: Image Deblurring m-file (deblur.m)

```

function deblur()

%Deblurring an image using Wiener filter

I=imread('1stdat.jpg');% read an 1st date and 2nd date image

%I=rgb2gray(I); % convert to grayscale

I=im2double(I); % convert to double type

F=fft2(I); % discrete fourier transfor

[ih iw]=size(I);%get size

% To estimate SNR

% take low freq part as signal

```

```

H=Butterworth(ih,iw,100,2,0); %get Butterworth filter

G=F.*fftshift(H); %filter in freq domain

g=ifft2(G); % inverse discrete fourier transform

img=abs(g); % get the absolute value

% take high freq part as noise

in=I-img;

%Signal power

sp=sum(sum(img.*img));

% Noise power

np=sum(sum(in.*in));

%signal to noisesignal ratio

snr=sp/np;

snr

H=Gaussian(ih,iw,50,0); %get blur spectrum which is Gaussian filter

H=fftshift(H);

Fdb=WienerFilter(F,H,1/snr); %deblur using Wiener filter

gdb=ifft2(Fdb); % inverse discrete fourier transform

idb=im2uint8(abs(gdb)); % convert grayscale

end

function Fd=WienerFilter(G,H,NSR)

AH2=H.*conj(H); % get magnitude square

Fd=( (AH2./(AH2+NSR)) ./H) .*G;

End

%-----End of -----deblur.m m-file-----
-----%

```

Listing 4: Image Sharpening (unsharp.m)

```

function imOut = unsharp(im, G, lambda)

% O = UNSHARP(IM,G,L) Unsharp masking

% Performs unsharp masking as follows:

% O = IM + L*G

% where G is the gradient performed by GRADIENTFILT(IM)

% O is returned normalized to (0,1)

%G = gradientfilt(im); imOut = im + (lambda * G);

imOut = imnorm(imOut);

```

Listing 5: Sobel Edge Detection m-file (sobel.m)

```

% 1st Date image

A=imread('C:\Users\tm16100806\Pictures\1stdate.jpg'); B=rgb2gray(A);
C=double(B);

D=rgb2gray(A);

for x=1:size(C,1)-2

    for y=1:size(C,2)-2

        %Sobel mask for x-direction:

        Gx=((2*C(x+2,y+1)+C(x+2,y)+C(x+2,y+2))-(2*C(x,y+1)+C(x,y)+C(x,y+2)));

        %Sobel mask for y-direction:

        Gy=((2*C(x+1,y+2)+C(x,y+2)+C(x+2,y+2))-(2*C(x+1,y)+C(x,y)+C(x+2,y)));

        %The gradient of the image

        %B(i,j)=abs(Gx)+abs(Gy);

        B(x,y)=sqrt(Gx.^2+Gy.^2);

    end

end

% figure,imshow(B); title('Sobel gradient');

```

```

% imwrite(B,'C:\Users\tml6100806\my_images\output\output1.jpg')

% 2nd Date image
E=imread('C:\Users\tml6100806\Pictures\2nddate.jpg');
F=rgb2gray(E); G=double(F); H=rgb2gray(E);
for x=1:size(G,1)-2
    for y=1:size(G,2)-2
        %Sobel mask for x-direction:
        Gx=((2*G(x+2,y+1)+G(x+2,y)+G(x+2,y+2))-(2*G(x,y+1)+G(x,y)+G(x,y+2)));
        %Sobel mask for y-direction:
        Gy=((2*G(x+1,y+2)+G(x,y+2)+G(x+2,y+2))-(2*G(x+1,y)+G(x,y)+G(x+2,y)));

        %The gradient of the image
        %B(i,j)=abs(Gx)+abs(Gy);
        F(x,y)=sqrt(Gx.^2+Gy.^2);
    end
end

% imwrite(F,'C:\Users\tml6100806\my_images\output\output2.jpg')

```

Listing 6: Image change detection m-code (xor_operator.m)

```

% Image change detection

% Exclusive-OR (XOR) operation on images
image_variable1= imread('C:\Users\tml6100806\Pictures\firstdate.jpg');

image_variable2= imread('C:\Users\tml6100806\Pictures\seconddate.jpg');

```

```

binary1=im2bw(image_variable1);
binary2=im2bw(image_variable2);

%output= xor(binary1, binary2);
output = bitxor(binary1, binary2);

% imwrite(output,'C:\Users\tm16100806\my_images\output\output.jpg')

function c = xor
A = "1stdate.jpg";
B = "2nddate.jpg";
C = "out.jpg";
(@(a,b,c) imwrite(imread(a)~=imread(b),c)) (A,B,C)

```

Listing 7: Image Contrast Stretching

```

%Contrast stretching image 1st date and 2nd date image
%img is the 1stdate and 2nd date images
function new =hist_extend(img)
h = imhist(img);
s = sum(h);
min=0;
acum = 0;
    while acum < 0.01*s
        min = min+1;
    end
    13
max

```

```

acum =0;

while acum

    max = max-1;

    acum = acum+h(max);

end

newimg = img-min;

newimg=(255/(max-min))*newimg

% -----end -----%

```

Listing 8: PSNR and MSE

```

I=imread('/Users/biustadmin/Desktop/code folder1/input.png');
ref=imread('/Users/biustadmin/Desktop/code folder1/ref.png');

I = imresize(I, size(ref));

n=size(I);

M=n(1);

N=n(2);

MSE = sum(sum((I-ref).^2))/(M*N);

PSNR = 10*log10(256*256/MSE);

fprintf('\nMSE: %7.2f ', MSE);

fprintf('\nPSNR: %9.7f dB', PSNR);

```