NINEVAH UNIVERSITY

COLLEGE OF ELECTRONICS ENGINEERING

COMPUTER AND INFORMATION ENG. DEPARTMENT

FPGA Implementation of Video Edge Detection Based on Fractional Order Differentiation

By

Ibtisam Edrees Kanaan

M.Sc. Thesis

In

Computer and Information Engineering

Supervised by

Dr. Emad Atiya Khalaf

Dr. Majid Dherar Younus

2021 A.D. **1442 A.H.**

FPGA Implementation of Video Edge Detection Based on Fractional Order Differentiation

A Thesis Submitted

By

Ibtisam Edrees Kanaan

To

The Council of the College of Electronic Engineering

Ninevah University

As a Partial Fulfillment of the Requirements

For the Degree of Master of Science

In

Computer and Information Engineering

Supervised by

Dr. Emad Atiya Khalaf

Dr. Majid Dherar Younus

2021 A.D. **1442 A.H.**

بسم الله الرحمن الرحيم

وعلمك ما لم تكن تعلم

وكان فضل الله عليك عظيما

صدق الله العظيم

من الآية 113 سورة

النساء

Supervisor's Certification

I certify that the dissertation entitled (**FPGA Implementation of Video Edge Detection Based on Fractional Order Differentiation**) was prepared by **Ibtisam Edrees Kanaan** under my supervision at the Department of Computer and Information Engineering, University of Ninevah, as a partial requirement for the Master of Science Degree in Computer and Information Engineering.

Signature:	Signature:
Name: Dr. Emad Atiya Khalaf	**Name: Dr. Majid Dherar Younus**

Department of Computer and Information Engineering

Date: / /2021

Linguistic Advisor Certification

I certify that the linguistic evaluation of this thesis entitled **"FPGA Implementation of Video Edge Detection Based on Fractional Order Differentiation'** was carried out by me and it is accepted linguistically and in expression.

Signature:

Name:

Date: / /2021

Post-Graduate Committee Certification

According to the recommendations presented by the supervisor of this dissertation and the linguistic reviewer, I nominate this dissertation to be forwarded to discussion.

Signature:

Name: Assistant Prof. Maan A. S. Al-Adwany

Committee Certification

We, the examining committee, certify that we have read this dissertation entitled (**FPGA Implementation of Video Edge Detection Based on Fractional Order Differentiation),** and have examined the postgraduate student (**Ibtisam Edrees Kanaan**) in its contents. In our opinion; it meets the standards of a dissertation for the degree of Master of Science in Computer and Information Engineering.

Signature:	Signature:	Signature:
Name: Assist Prof.	**Name: Assist Prof.**	**Name: Assist Prof.**
Dr.Mohammed H. AL-Jammas	**Dr.Basma MK.Younis**	**Dr.Sinan Alkassar**
Head of committee	Member	Member
Date: / /2021	**Date: / /2021**	**Date: / /2021**

Signature:	Signature:
Name:Dr. Emad Atiya Khalaf	**Name:Dr. Majid Dherar Younus**
Member and Supervisor	Member and Supervisor
Date: / /2021	**Date: / /2021**

The college council, in its ………… meeting on / /2021, has decided to award the degree of Master of Science in Computer and Information Engineering to the candidate.

Signature:

Name:

Dean of the College

Date: / /2021

<u>Department Head Certification</u>

I certify that this dissertation was carried out in the Department of Computer and Information Engineering. I nominate it to be forwarded to discussion.

Signature:

Name: Assistant Prof. Maan A. S. Al-Adwany

Date: / /2021

ACKNOWLEDGEMENTS

Praise be to ALLAH, Lord of the World. Praise be to ALLAH, who gave me strength and success during the completion of this study.

I would like to express my sincere gratitude and thanks to my supervisors, **Dr. Emad Atiya Khalaf and Dr. Majid Dherar Younus** for their continuous guidance, valuable suggestions and constant encouragement throughout this work.

Thanks are due to the Dean of the Electronics Engineering College for his valuable assistance. My appreciation is extended to the Head and all the members of the Computer and Information Engineering Department for their support and assistance.

Also, I would like to thank my parents, who did not leave me feel alone during my study through their continuous supplications. Also, I would like to extend my sincere thanks and gratitude to all the members of my

family and my husband for their encouragement, support, and patience during my postgraduate studies.

.

Researcher

Ibtisam Edrees Kanaan

2021

Abstract

Edge detection is one of the vital research issues and a very important key step towards the realization of image features, image segmentation, and pattern recognition. It is a relatively old concept that relies on integer derivatives to detect image edges, mainly of the first or second-order derivatives. With the emergence of fractional calculus, edge detection is revisited with non-integer order derivatives. The present study proposes video edge detection based on five fractional-order Sobel operators. The design of the fractional-order Sobel filters based on Yi_Fei-1 to Yi_Fei-5 operators.

Graphical User Interface (GUI) has been developed in Matlab 2017b to analyze the proposed fractional and classical Sobel filters. Edge accuracy is one of the edge detector challenges; therefore supervised assessment for the proposed filters and classical one has been performed by using Mean Square Error, Symmetric Distance Measure, Relative Distance Error, Misclassification Error, Complemented Performance Measure and Complemented F Measure. Many images with their ground truths have been used in the evaluation. The results showed that fractional-order Sobel filter-based on Yi_Fei-2 outperforms on classical Sobel and other proposed filters.

Fractional-order Sobel filter based on Yi_Fei-2 has been implemented by using FPGA. Xilinx Vivado 2018.2 has been used in the aggregation with Matlab 2017b. HDL verifier through FPGA in the loop and Co-simulation methodology has been utilized in the FPGA implementation on Artix-7 NEXYS 4 kit. The root mean square error between hardware implementation and software simulation is 0.002.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

Abbreviation	Name
API	The Windows Application Interface Function
BER	Bit Error Rate
DR	Detector Result Edge Map
EDK	Embedded Development Kit
FIL	FPGA in the Loop Simulation and Verification
filWizard	FPGA in the Loop Wizard Matlab Function
FM	The Complemented F Measure
FN	False Negative Point
FOC	Fractional Order Calculus
FOM	Figure of Merit
FP	False Positive Point
FPGA	Field Programmable Gate Array hardware
FS_Yi_Fei-1	Fractional Sobel Filter Based Yi_Fei-1 Operator
FS_Yi_Fei-2	Fractional Sobel Filter Based Yi_Fei-2 Operator
FS_Yi_Fei-3	Fractional Sobel Filter Based Yi_Fei-3 Operator
FS_Yi_Fei-4	Fractional Sobel Filter Based Yi_Fei-4 Operator
FS_Yi_Fei-5	Fractional Sobel Filter Based Yi_Fei-5 Operator
G-L	Grümwald-Letnikov Definition
GT	Ground Truth Image
GUI	The Graphical User Interface
H/W	Hardware System
HDL	The Hardware Description Language
HIL	The Hardware in the Loop Methodology
LoG	Laplacian of Gaussian
ME	The Misclassification Error Metric
MRI	Magnetic Resonance Imaging

MSE	Mean Square Error
PM	The Complemented Performance Measure
PSNR	Peak Signal to Noise Ratio
RDEL	Relative Distance Error Measure
R-L	Riemann-Liouville Definition
SDM	Symmetric Distance Measure
SNR	Signal to Noise Ratio
Th	The High Threshold of Canny Operator
Tl	The Low Threshold of Canny Operator
TN	True Negative Point
TP	True Positive Point
XSG	Xilinx System Generator
Yi_Fei-1	The First Operator of Yi-Fei Pu
Yi_Fei-2	The Second Operator of Yi-Fei Pu
Yi_Fei-3	The Third Operator of Yi-Fei Pu
Yi_Fei-4	The Fourth Operator of Yi-Fei Pu
Yi_Fei-5	The Fifth Operator of Yi-Fei Pu

LIST OF SYMBOLS

Symbols	Name
${}_{x}D_{w}^{p}$	Major Operator of Fractional Calculus
$\binom{p}{k}$	Binomial Coefficient
$\bar{G}$	The Mean of Gradient Magnitude
$M2_x$	Fractional Sobel Filter Based Yi_Fei-2 Mask in x-direction
$M2_y$	Fractional Sobel Filter Based Yi_Fei-2 Mask in y-direction
$M3_x$	Fractional Sobel Filter Based Yi_Fei-3 Mask in x-direction
$M3_y$	Fractional Sobel Filter Based Yi_Fei-3 Mask in y-direction
$M4_x$	Fractional Sobel Filter Based Yi_Fei-4 Mask in x-direction
$M4_y$	Fractional Sobel Filter Based Yi_Fei-4 Mask in y-direction
$M5_x$	Fractional Sobel Filter Based Yi_Fei-5 Mask in x-direction
$M5_y$	Fractional Sobel Filter Based Yi_Fei-5 Mask in y-direction
$\|.\|$	The Number of Elements in a Set
$\emptyset$	The Gradient Angle
$\cap$	Intersection symbol
$\cup$	Union symbol
$\ulcorner$	The Complement of the Set
$\boldsymbol{D}_{DR}(pl)$	The Minimal Euclidean Distance Between *pl* and *DR, pl* belonging to GT
$\boldsymbol{D}_{GT}(pl)$	The Minimal Euclidean Distance Between *pl* and *GT, pl* belonging to DR
F	The Source Image
G_x	The Gradient in x-direction
G_y	The Gradient in y-direction
H	The Calculation Step in Image Processing

K	The Mask Size
L	Parameter in SDM and RDEL
M	Magnitude of Gradient for LoG Operator
O	Orientation of Gradient for LoG Operator
Pl	A Pixel in GT or DR
$\mathbf{R_x}$	Roberts Mask in x-direction
$\mathbf{R_y}$	Roberts Mask in y-direction
$\boldsymbol{S_x}$	Sobel Mask in x-direction
$\mathbf{S}_y$	Sobel Mask in y-direction
G	Over all Gradient in x and y direction
n	Integer Number
p	Fractional Order
$\boldsymbol{P_x}$	Prewitt Mask in x-direction
$\boldsymbol{P_y}$	Prewitt Mask in y-direction
α	The Value of Denominator in Masks Coefficients
β	A Real Number Used in Complemented F Measure

CHAPTER ONE

Introduction

1.1 Introduction

Edge detection is one of the commonly used techniques for extracting features of an image. The main purpose of edge detection is to reduce the quantity of the image data that has to be processed.Therefore, edge detection is considered as a fundamental stage in various image processing applications [1].

The abrupt changes of discontinuities in an image are defined as edges. Edges of an image are treated as a kind of vital information that can be separated by using detectors with different approaches. Edge detectors based on integer order derivative can be principally categorized into two groups: laplacian-based and gradient-based, which means the use of the second-order derivative and first-order derivative respectively [2,3]. The first-order derivative techniques usually result in thicker edges, which cause loss of image details. The second-order derivative techniques have a robust response to fine detail, but they are very sensitive to noise [4].

In recent years, the fractional-order calculus has been involved in the edge detection techniques. Since the characteristic of the fractional-order derivative can preserve nonlinearly more low-frequency contour features in those smooth areas, keep high-frequency marginal features where grayscale changes frequently, and also enhance medium-frequency texture details [5]. Then, to certify that an edge detector is reliable, it must be severely evaluated before being used in a computer vision application. Hence, a supervised evaluation measure calculates a dissimilarity criterion between a detector result and a ground truth edge map, commonly got from expert judgment or synthetic data [6 - 10].

In image processing applications, the Field Programmable Gate Array hardware (FPGA) is being more and more significant, therefore it can be utilized in the practical implementation of any digital logic function. FPGA hardware has been considered as a substitution for the implementation of the software algorithms. The presentation of FPGA has a significant impact on image or video processing. This is due to the FPGA's ability to provide parallel and high computational density, compared to the general purpose of the microprocessor, and the flexibility in the development of image processing algorithms on FPGA because of its capability of re-programmable actuality[11].

1.2 Problems Statement

Many approaches based on integer order derivative have been used to detect the edges in an image, but a propose edge detector is necessary to produces thin edges and has the maximum number of true edges and the minimum number of missing edges that form the borders of the shapes in the image, additionally, it has the minimum distance of the incorrect edges (fake and missing edges) from their actual position.

1.3 Related Work

Many researchers have applied many approaches to detect the edge in the image and video processing, assessed it to get the best one and some are presented the hardware implementation of edge detector:

1.3.1 Edge Detection Based on Integer Order Derivative

In 2013 D. Poonam and N. Sahu [12] demonstrated a study on the most commonly utilized edge detection methods such as Roberts, Sobel, Prewitt, Laplacian of Gaussian, and Canny. The most important thing that has been noticed was that the canny operator had the best results. However, the Laplacian of Gaussian presented similar results; but with missing edges comparing to the Canny detector. Roberts, Prewitt, and Sobel's resulted in actually different from those of Laplacian of Gaussian and Canny.

In 2015 Ş. Öztürk and B. Akdemir [13] presented some of the general edge detection approaches such as Prewitt, Sobel, Roberts, LoG, and Canny which were applied to glass texture analysis, their endeavor aimed to define glass surface defect by using the same image. Comparison has been performed between detectors. It has been found that LoG operator is superior to other operators in defining texture analysis.

In 2016 L. Bhargav et al. [14] discussed three edge segmentation techniques; Sobel operator, Canny operator, and LoG operator, and they made a comparison between the results of the operators to choose the best one. It has been found that Sobel operator outperformed among the other filters.PSNR parameter was used in performance evaluation.

In 2017 K. B. Krishnan et al. [15] analyzed and compared the performance of various edge detection methods like Canny optimal, Sobel, Prewitt, and LoG detectors. Canny acts as an optimal technique for edge detection because of its superior performance in complex images or noise images in comparison to other methods.

1.3.2 Edge Detection Based on Fractional Order Differentiation

In 2009 Y. Pu et al. [16] proposed six fractional differential masks and presented individually the structures and parameters of all masks in the eight directions. Experiments showed that, for a digital image that is rich in details, the capability of fractional differential-based approaches seemed obviously better than the classical methodologies.

In 2015 D. Tian et al. [4] proposed a novel fractional-order gradient detector to extract features from medical image structures. The proposed operator is considered as a generalization of the integeral Sobel operator based on the G-L definition of non-integer order derivative. The experiments showed that the modified Sobel operator yields well observable effects.

In 2016 S. M. Ismail et al. [17] presented a comparative study of four fractional-order filters used for edge detection. The four filters are based on G-L definition with different modifying. All of them are of size 3x3. It were used to extract the structural features of two groups of medical brain images of Alzheimer disease patients. The comparison parameters that used MSE, execution time, SNR, and PSNR parameters. Two types of noises were added to the original image, random Gaussian noise, and Salt and Pepper noise. The performance of filters is explained to be variable depending on the noise type which was added to the image.

In 2017 W. S. ElAraby et al. [18] presented a comparative investigation of edge detection techniques based on several fractional-order filters. The first two operators present several fractional masks to detect edges. Then, the two other operators employ a genetic algorithm to enhance edges detected from the preceding fractional masks. The performance evaluation was done based on the peak signal to noise ratio (PSNR) and bit error rate (BER). It showed that non-integral edge detection included genetic algorithm provides better edges compared to the traditional approaches.

In 2018 R. John and N. Kunju [19] presented a comparative survey of fractional order operator and a first-order operator on MRI Images. The results explained that the integer-order filter was outperformed by the fractional filter. It can be predictable to offer good effects in a lot of applications where the traditional filter is used. Quality assessment was performed in terms of PSNR and MSE.

In 2019 D. Li et al. [20] proposed a novel fractional-order detector, united compound derivative operator, and Crone operator. In a complex derivative operator, a one-dimensional fractional order integral filter was transformed into a two-dimensional fractional order integral filter, and a two-dimensional differential kernel was built by expanding the horizontal and vertical components of the differential template of the Crone operator. The experimental results showed that, for all kinds of noise and non-noise images, this operator can obtain an objective detection with clear edges and perfect structures. Detection selectivity and noise tolerance are enhanced compared to the Crone operator and composite derivative operator, and edge enhancement and structure preservation are realized.

In 2019 C. Yaacoub and R. A. Z. Daou [21] proposed the construction of a fractional-order Sobel detector. Traditional Sobel operator was used for the first-order derivative, and fractional calculus was invited for non-integer order. In the sense of road obstacle detection, the proposed method was introduced, tested, and contrasted with the traditional Sobel edge detector.

In 2019 M. Mekideche and Y. Ferdi [22] proposed and then examined a short edge detector method without denoising operation based on a steerable order mask used for edge enhancement. It has been shown that the proposed algorithm has the capability of detecting edges and immunity to noise, which means that the smoothing pre-process is not very necessary with the fractional detector and it can be used instead of classical ones to eliminate the smoothing processing.

In 2020 J. E. Lavín-Delgado et al. [23] proposed a novel fractional kernel core comprised generalizing the classical Sobel operator using Caputo–Fabrizio fractional definition without the need to a singular kernel significant. The results illustrated the effectiveness of the novel detector for finding both weak edges (texture) and strong edges (object outlines). Also, it was verified the capability of the new fractional detector to robust the noise.

In 2020 M. Hacini et.al. [24] presented a new operator created for two-dimensional non-integer order differentiation, a proposed operator based on the one-dimensional extension of Charef fractional differentiation. A new multi-directional kernel was advised and a new adaptive fractional-order calculation using the properties of gradient computation was introduced. It has been used in de-noising and edge detection problems using synthetic and real images. Due to its low complex computation, the obtained results prove its validity. FOM, MSE, and PSNR were used to assess the performance of the proposed kernel for feature extraction.

1.3.3 Performance Evaluation

In 2017 H. Abdulrahman et al. [10] illustrated a semi-automatic way to label ground truth data and detailed numerous supervised edge detection assessments and applied them toward a qualitative assessment. Experiments demonstrated the importance of choosing the precise ground truth map, where an imprecise ground truth edge map concerning edge localization penalizes the accurate edge detectors and/or benefits of the imprecise algorithms.

In 2017 D. Sadykova and A. P. James [25] provided a survey of all the most significant evaluation metrics which can be utilized for the benchmark of the qualitative performance for result edge images. Categorizing four main sets of metrics and also provided a significant perception into the assessment protocol and major equations.

In 2018 B. Magnier et al. [9] presented a review on reference edge detection assessment methods to measure the differences between a ground truth edge map and a candidate, calculated by performance metrics and suggested a new qualitative metric. The new measure gave a whole assessment of the quality of an edge map by considering the number of false negatives and false positives, and the grades of shifting. Experiments demonstrated that the required objects are not always entirely obvious by using incorrect assessment measurements.

1.3.4 Hardware Implementation

In 2012 Y. Said et al. [26] presented architecture of Sobel Filter for edge detection using Xilinx System Generator. The design was implemented targeting a Spartan3A DSP 3400 device (XC3SD3400A-4FGG676C) then a Virtex 5 (xc5vlx50-1ff676). The results shown that the implemented Sobel edge detector architecture using low cost available Spartan 3 development system with Xilinx chip XC3S50 -5PQ208 has 54.505 MHz maximum frequency and uses 177 CLB slices with 23% utilization.

In 2013 A. M. Khidhir and N. Y. Abdullah [11] presented FPGA implementation of Sobel detector through using Virtex-5 ML506 board to extract the edges from grayscale images. In the beginning, the standard Sobel detector was utilized to find the edges from grayscale images. Then, the Sobel detector was modified to extract the edges with image denoising. The system for edge detection was performed with the incorporation of Matlab environments and Embedded Development Kit (EDK). It has been found that the edge map resulted from the modified Sobel detector was better than the classical one.

In 2014 P. K. Dash et al. [27] implemented Sobel and Prewitt technique to detect edges in video and image processing purposes through using a Model-Based Approach and FPGA, by using ALTERA DE2 FPGA kit, the whole video edge detection system was implemented on CYCLONE II FPGA hardware.

In 2015 J. C. T. Hai et al. [28] presented an alternative approach using a model based design framework based on HDL Coder, Vision HDL Toolbox and Simulink to accelerate the design of video and image solution.

The design was implemented in an Altera DE2-115 FPGA board. The results shown that the automatic code generation from HDL Coder and high level of abstraction allowed designer to focus on algorithm design instead of coding, thus saving valuable project development time.

In 2017 M. Alareqi et al. [29] demonstrated optimization and developing a real-time Hardware Co-Simulation of edge detection system with an input coming from a live video acquired from a digital camera, and outputs were displayed on a video display and verified the video results in real time. The system was implemented on Virtex-5 XUPV5-LX110T FPGA, Verification on Xilinx Virtex-5 XUPV5-LX110T development board was indicated that the system can accurately detect the image edge and satisfy requirements of the real-time video image edge detection by providing minimum hardware resources, low power consumption, and high image quality in terms of improved picture signal to noise ratio (PSNR).

In 2019 O. H. Moustafa and S. M. Ismail [30] presented the FPGA implementation of edge detection by using fractional order filters. The design of the offered hardware contributes the single-precision floating-point arithmetic, IEEE 754 representation, to have accuracy in implementation and applied, on VIRTEX 5 FPGA kit, the amount of clock frequency is 170 MHz. The fractional filters are utilized to detect edges of an MRI scan as well as standard images.

In the present work. The classical Sobel filter will be generalized based on five fractional operators proposed by [16], the five fractional Sobel filters that will be proposed and the classical one should be evaluated based on supervised assessment then implement the best filter on FPGA.

1.4 Thesis Objectives

The present study aims at investigating the following points:

1. Study of video edge detection algorithms.
2. Study of fractional calculus and its application in edge detection.
3. Develop the required software to implement and examine the proposed fractional-order operators.
4. Certify the proposed fractional operators based on supervised assessment.
5. Implement of the proposed fractional-order video edge detection system on FPGA.

1.5 Thesis Layout

The techniques of edge detection based on integer, non-integer calculus, and supervised evaluation metrics are presented in chapter two. Chapter three introduces the design of an edge detection system using fractional-order Sobel operators based on Yi_Fei-1 to Yi_Fei-5, the software implementation, and results analysis. Whereas the FPGA implementation of the proposed video edge detection system is demonstrated in chapter four. Finally, conclusions and suggestions for further research are proposed in chapter five.

CHAPTER TWO

Edge Detection Techniques and Performance Evaluation Metrics

2.1 Introduction

In the present chapter, some of the classical edge detection approaches have been illustrated, the most famous definitions of fractional order calculus, and then demonstrated some methods to assess the edge detector.

2.2 Edge Detection

Image edge detection is one of the preliminary standards in pattern recognition applications and computer vision [1]. In image processing, edge detection is considered as one of the vital research works and an important key step towards the realization of image features. Therefore, other image processing applications such as identification, segmentation, and object recognition can occur when edges of an object are detected [31].

An edge is known as a discontinuity. The discontinuities may be represented in terms of gray level, color, or texture of the image. Edge detection aims at extracting significant information about the objects existing in an image, such as illumination, geometry, and reflectivity [32]. For example, the pixel's gray-level whose value is identical to the other adjacent pixel’s gray-level; there is maybe not an edge at that position. Actually, if the neighbors of a pixel have widely varying gray levels, the pixel will possibly be an edge pixel [33].

All available edge detection operators that are targeted to be sensitive to certain types of edges depending on the variables involved in the choice of an edge detection operator which comprises[32]:

- **Edge Orientation:** The characteristic of the operator direction in which the operator is most sensitive to the edges limited by the geometry of the operator. It can be used to detect horizontal, vertical, or diagonal edges.
- **Noise Environment:** In noisy images, the edge detector result is different, because both edges and noise have high-frequency content. Trying to minimize the noise effect in distorted and blurred edges, operators must be larger in scope so that they can cover enough data to discount localized noisy pixels. This, indeed, leads to minimizing the precise localization of the isolated edges.
- **Edge Structure:** Edges are not limited to a step change in intensity, but there is a gradual change in intensity as a result of poor focus or reflection. So operator must be sensitive to a gradual change, hence it does not have problems of true edge detection, false edges, missing, edge localization, and high computational time.

2.3 Classical Edge Detection Techniques

Many ways are available to perform edge detection. However, the classical edge detector can be grouped into two main categories:

2.3.1 Gradient-Based Techniques

The gradient-based is the most well-known technique for the edge detection. It detects the edges by finding minimum and maximum differentiation in the first-order derivative of the image. All the gradient-based methods have convolution masks that calculate the strength of the slope in directions that are orthogonal to each other, usually, vertical and horizontal. Subsequently, the influences of the various components of the slopes are aggregated to grant the overall value of the edge strength [33, 34]. Gradient-based edge detectors are:

- **Roberts Operator**

  It is one of the earliest approaches for edge detection, in which the image intensity is computed according to the first order using two 2x2 convolution masks in a 2x2 neighborhood of the subpixel of interest. The high sensitivity to noise is the main drawback of the Roberts Operator. The common masks are given by $\boldsymbol{R_y}$ and $\boldsymbol{R_x}$ [23, 32,34].

$$\boldsymbol{R_y} = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix} \qquad \boldsymbol{R_x} = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$$

  Using equation (2.1) and equation (2.2) to get the magnitude of the gradient.

$$G_x = f \cdot \boldsymbol{R_x} \quad , \qquad G_y = f \cdot \boldsymbol{R_y} \qquad (2.1)$$

$$|G| = |G_x| + |G_y| \qquad (2.2)$$

Where f is an image,

$G_{x,}\, G_y$ are the gradients of f in x, y-direction respectively.

- **Sobel Operator**

  It contains two 3×3 convolution masks to obtain the gradient in two directions (i.e. horizontal and vertical orientation). To calculate the gradient component in both directions, the mask is convolved over an image separately. The Sobel Operator uses the common masks which are specified by $\boldsymbol{S_y}$ and $\boldsymbol{S_x}$.

$$\boldsymbol{S_y} = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \qquad \boldsymbol{S_x} = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}$$

  Equation (2.3) shows the convolution of input image f with the horizontal mask ($\boldsymbol{S_x}$) and equation (2.4) shows convolution of the image with the vertical mask ($\boldsymbol{S_y}$).

$$G_x = \{f(x+1,y-1) + 2f(x+1,y) + f(x+1,y+1)\} - \{f(x-1,y-1) + 2f(x-1,y) + f(x-1,y+1)\} \quad (2.3)$$

$$G_y = \{f(x-1,y-1) + 2f(x,y-1) + f(x+1,y-1)\} - \{f(x-1,y+1) + 2f(x,y+1) + f(x+1,y+1)\} \quad (2.4)$$

Where, $f(x,y)$ is an integer pixel coordinate's value of the input image$f$.

The two ways of calculating the magnitude of the gradient are the Euclidean distance equation (2.5) and the city block distance (Manhattan) equation (2.6) [35,36].

$$|G| = \sqrt{(G_x{}^2 + G_y{}^2)} \quad (2.5)$$

$$|G| = |G_x| + |G_y| \quad (2.6)$$

The gradient angle is given by:

$$\emptyset = \tan^{-1} G_y/G_x \quad (2.7)$$

- **Prewitt Operator**

It seems relatively the same as the Sobel operator with the dissimilarity of the constant value (C). Prewitt Operator provides a better performance on horizontal and vertical edges in the images and higher responses for noisy images. It is a widely used method to calculate the magnitude (G) of the edges [37,38].

$$G = \sqrt{(G_x{}^2 + G_y{}^2)} \quad (2.8)$$

The Gx, Gy partial derivation are given in equation (2.9) and equation (2.10) respectively.

$$G_x = (a2+Ca3+a4) - (a0+Ca7+a6) \quad (2.9)$$

$$G_y = (a0+Ca1+a2) - (a6+Ca5+a4) \quad (2.10)$$

Where C=1 and a0, a1 …….a7 are:

$$a0 = f(x-1, y-1), \tag{2.11a}$$

$$a1 = f(x, y-1), \tag{2.11b}$$

$$a2 = f(x+1, y-1), \tag{2.11c}$$

$$a3 = f(x+1, y), \tag{2.11d}$$

$$a4 = f(x+1, y+1), \tag{2.11e}$$

$$a5 = f(x, y+1), \tag{2.11f}$$

$$a6 = f(x-1, y+1), \tag{2.11g}$$

$$a7 = f(x-1, y) \tag{2.11h}$$

So the common kernel cores of the Prewitt Operator are:

$$\boldsymbol{p_x} = \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix} \qquad \boldsymbol{p_y} = \begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix}$$

2.3.2 Gaussian Based Techniques

This technique finds edges by detecting the zero crossings in the second-order derivative of an image (the maximum difference point in first order represent zero crossing point in second order). The Gaussian based techniques are the following:

- **LoG Operator**

  This method is used to calculate the second-order derivate edges of an image. The laplacian operation is utilized to detect the edges either in the dark area or light area.

  By the end of the laplacian process, the image is applied to a Gaussian filter to minimize the unwanted noisy pixels though the edge location is identified accurately. The gradient of laplacian for a two-dimensional image is given in equation (2.12)

$$G = \frac{\partial f^2}{\partial x^2} + \frac{\partial f^2}{\partial y^2} \tag{2.12}$$

Then, the resulting convolution kernel is applied to the image. The masks that can be used to find the magnitude of the gradient(M)and orientation(O) in the Laplacian of Gaussian gradient edge detector are shown below [39,40].

$$O = \begin{bmatrix} -1 & 2 & -1 \\ 2 & -4 & 2 \\ -1 & 2 & -1 \end{bmatrix} \qquad M = \begin{bmatrix} 1 & 1 & 1 \\ 1 & -8 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

- **Canny Operator**

A canny algorithm is considered as an optimal algorithm for edge detection, and it is the most commonly used in practice. The most important criteria of canny related to catch low error rate and the location of detected edges should be precise and only one point must be returned by the detector for each true edge (i.e. the number of local maxima around the edge must be minimum).

A canny operator consists of a few simple steps. The first step is image denoising. Because the noise leads to incorrect localization of the object's boundaries in the image, it must be smooth. In the second step, masking in which the image intensity is derivate using two-dimensional kernels which is taken using an edge detection operator such as Prewitt, Sobel, then calculating the gradient of the image based on (G_x, G_y) results. $\boldsymbol{S_y}$, $\boldsymbol{S_x}$ a 3x3 mask pair that can be utilized for canny algorithms respectively.

$$\boldsymbol{S_x} = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} \qquad \boldsymbol{S_y} = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$$

equation (2.13) shows the formula used to calculate the overall gradient.

$$|G| = |G_x| + |G_y| \tag{2.13}$$

The fourth step is non maximum superission which determines the direction of edge regions by looking at the x and y directions of the calculated gradient. And the direction of the gradient is specified by checking the pixels at definite angles (often 0°, 45°, 90°, and 135°), then suppressing the non-maximum points (i.e. non-edge pixels are eliminate to 0 levels). Finally, the image is thresholded [41, 42].

Hysteresis threshold is based on two thresholds to detect the edges. The high threshold (th) and the low threshold (tl) are worked to find definite and weak edges successively. The magnitude of the derivative for a pixel detected as the edge compared with th if it is greater, then, it seeks in the direction of the derivative and if the derivative magnitude in the succeeding pixel is greater than tl, that pixel is also defined as an edge. This tracking is nonstop until the opposite pixel with a derivative magnitude is becomes less than tl [32, 12]. The flow chart of the canny detector is shown below:

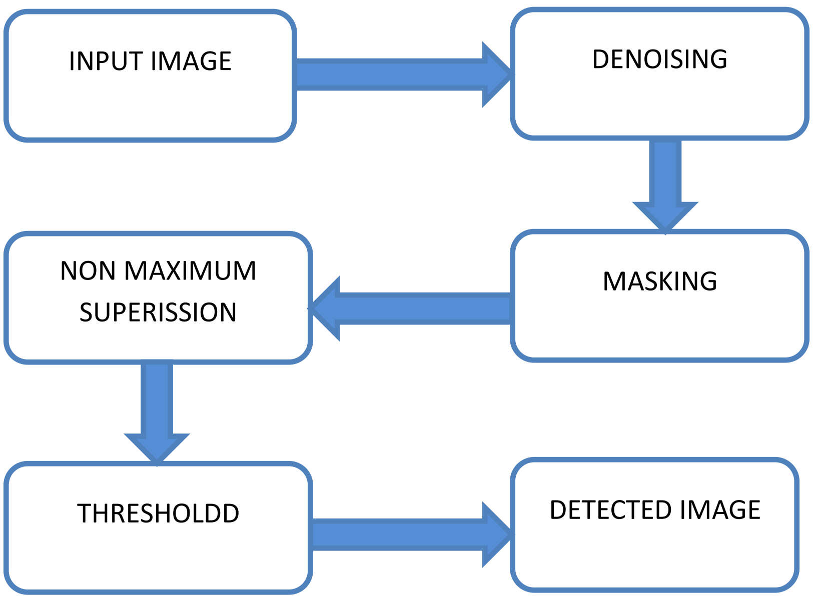


Figure (2.1): Canny operator steps

2.4 Fractional Calculus Concept

Recently, Fractional calculus has been considered as a special interest in several scientific fields and applications since it can describe models or physical methods more accurately than any traditional form of integer order [23, 43].

Fractional calculus, which is also named as non-integer order calculus, is a generalization of the classical (integer-order) calculus. The conception of the derivative is usually related to an integer; assumed a function, one can differentiate it one, two, three times, and thus. In the same context, the ability to differentiate a function, a real number of times, known as the fractional order calculus [24,44].

Since several researchers in various areas have been investigating fractional differentiation. Riemann-Liouville (R-L), Grümwald-Letnikov (G-L), and Caputo have also formed the most commonly accepted meanings [24, 43]. Fractional order calculus (FOC) is a generalization of derivation and integration to non-integer order with a major operator as shown in equation (2.14).

$$
{}_xD_w^p = \begin{cases} \int_x^w d(t)^p\,, & p < 0, \\ 1, & p = 0, \\ \frac{d^p}{dt^p}\,, & p > 0. \end{cases} \tag{2.14}
$$

Where p the fractional-order which is a real number, [x, w] is the interval time and t is the time.

In image processing, the fractional-order workflow is shown in figure (2.2). It holds three steps. Firstly, the active operator, model, or equation including usual integration and differentiation is chosen. Subsequently, the usual differentiation and integration are generalized to fractional order (steerable order) using the fractional calculus definition (G-L, R-L, or Caputo). Finally, discretization methods will be used to calculate a numerical approximation of the fractional-order operator, model, or equation [5].

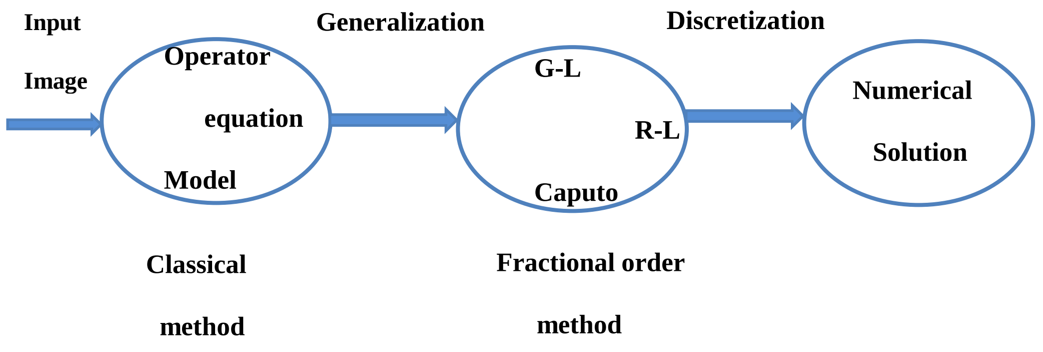


Figure (2.2): Workflow of FOC in image processing [5].

2.5 Fractional Calculus Definitions[42].

In contrast to integer calculus, till now, the fractional derivative does not have a standardized definition. The famous definitions of fractional calculus are:

- **Fractional Derivative G–L Definition:**

$${}_xD_w^pI(t) = \lim_{h\to 0} \frac{1}{h^p} \sum_{k=0}^{\left[\frac{t-x}{h}\right]} (-1)^k \binom{p}{k} I(t-kh) \tag{2.15}$$

Where $\binom{p}{k}$ is a binomial coefficient which calculates by Equation (2.16)

$$\binom{p}{k} = \frac{(p)(p-1)\ldots(p-k+1)}{k!} \tag{2.16}$$

And (h=1)is the calculation step in image processing, k represents the size of the mask which must be equal to or greater than 3, and [x w]is the interval time, p is a real number(Fractional Order) and n is an integer number.

- **Fractional Derivative R–L Definition:**

  The Riemann–Liouville fractional derivative is:

$$ {}_{x}D_{w}^{p}I(t) = \frac{1}{\Gamma(n-p)}\left(\frac{d}{dt}\right)^{n}\int_{x}^{w}\frac{I(\tau)}{(t-\tau)^{p-n+1}}d\tau \ ,(n-1) < p < n \qquad (2.17)$$

Where $\Gamma\,(n) = \int_{0}^{\infty} \mathrm{t}^{\mathrm{n}-1}\,\mathrm{e}^{-\mathrm{t}}\mathrm{dt}$ It can be generalized as

$\Gamma\,(n) = (n-1)!$ Which is the gamma function.

- **Fractional Derivative Caputo Definition:**

  The Caputo fractional derivative is:

$$ {}_{x}^{C}D_{w}^{p}\,I(t) = \frac{1}{\Gamma(n-p)}\int_{x}^{w}\frac{I^{(n)}(\tau)}{(t-\tau)^{p+1-n}}d\tau, \quad (n-1) \le p < n \qquad (2.18)$$

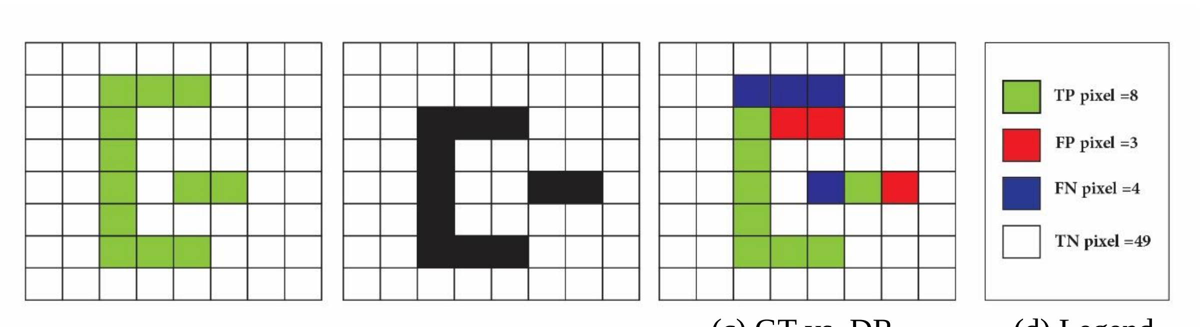
Where p is a real number and n is an integer number.

2.6 Evaluation Metrics

The edge detector should be strictly evaluated before being used in a computer vision tool. To verify the reliability of an edge detector, a dissimilarity evaluation computes a score between a ground truth edge map and a detector result image. Various techniques have been developed to evaluate the desired edge map. Several methods are based on the number of true positive, true negative, false positive, and/or false negative points denoted as error measures involving only statistics. Many assessment methods calculate the distance from the location where an edge point should be positioned denoted as an assessment involving distances of misplaced pixels [9,10].

2.6.1 Error Measures Involving Only Statistics

The confusion matrix forms a cornerstone in edge detection assessment techniques. Let GT be the Ground truth edge map and DR the result of the edge detector. By Comparing pixel per pixel of GT and DR, the shared existence of edge/non-edge points is the first measure to be evaluated. All points are divided into four groups as illustrated in figure (2.3).



(a)GT 8X8 (b) DR 8X8 (c) GT vs. DR 8X8 (d) Legend of (c)

Figure (2.3): An example of a comparison between the result edge map (*DR*) and ground truth image (*GT*).

Where

- False Positive points (*FPs*), fake detected edges of *DR:*
  FP = [⌐*GT* ∩ *DR*],
- False Negative points (*FNs*), missing edge points of *DR:*
  FN = [*GT* ∩ ⌐*DR*],
- True Negative points (*TNs*), mutual non-edge points:
  TN = [⌐*GT* ∩ ⌐*DR*],
- True Positive points (*TPs*), mutual edge points of *GT* and *DR:*
  TP = [*GT*∩ *DR*].

([.]) denotes the number of elements in a set, (⌐) is the complement of the set, and (∩) is an intersection symbols.

Various edge detection assessments containing confusion matrices are discussed in [9, 10]. The Misclassification Error (*ME*), Complemented Performance Measure (*PM*), and Complemented F Measure (*FM*) will be used as a criterion containing confusion matrices.

- **Misclassification Error** mirrors the percentage of “True Negative” pixels wrongly assigned to “True Positive”, and vice versa, “True Positive” pixels wrongly assigned to “True Negative” pixels [9, 10, 45]. The *ME* varies from 0 for the best edge map to 1 for the worst edge map image as shown in equation (2.19).

$$ME = 1 - \frac{TN+TP}{TP+TN+FN+FP} \tag{2.19}$$

- **Complemented Performance Measure** The performance measure is a scalar taking values between 0 and 1. If all *TP* pixels are detected and no falsely pixels are detected as edge pixels, then (*PM*=0). For all other cases, *PM* takes unified values, which are closer to 1 as more edge pixels are missed and/or incorrectly detected by the edge detector [46].

$$PM = 1 - \frac{TP}{TP+FN+FP} \tag{2.20}$$

- **Complemented F Measure**

$$FM = 1 - \frac{\frac{TP}{TP+FP} \times \frac{TP}{TP+FN}}{\beta \times \left(\frac{TP}{TP+FP}\right) \times (1-\beta) \times \left(\frac{TP}{TP+FN}\right)} \tag{2.21}$$

Where β a real number $\in[0, 1]$, FM does not consider the “True Negative” pixels, which are dominant in detector result images. The complement of *FM* measure translates a value close to 0 as the best edge map[47].

- **Mean Square Error**

$$MSE = \frac{1}{MN} \sum_{i=1}^{M} \sum_{j=1}^{N} \left(GT_{i,j} - DR_{i,j}\right)^2 \tag{2.22}$$

The *MSE* is also used as an error measure to represent the cumulative squared errors between the edge map of the proposed edge detector (*DR*) and the ground truth image (*GT*) [23, 42]. The minimum difference between *GT* and *DR* is being when *MSE* close to 0.

2.6.2 Assessment Involving Distances of Misplaced Pixels

A supervised edge map quality measure involves that an emigrant edge must be penalized in function not limited to FNs and/or FPs, but also of the distance from the location where it must be located. The most important criteria including distances illustrated in [48, 49].

So, for a pixel *pl* belonging to the detector result *DR*, $\boldsymbol{D}_{GT}$ (*pl*) is the minimal Euclidean distance between *pl* and *GT*. If *pl* belongs to the ground truth *GT*, $\boldsymbol{D}_{DR}$ (*pl*) is the minimal distance between *pl* and *DR*. The coordinates of two pixels *pl* and *o*, mathematically, representing (x_{pl}, y_{pl}) and *(xo, yo)*, respectively; hence, $\boldsymbol{D}_{GT}(pl)$ and $\mathbf{D}_{DR}(pl)$ are described by:

$$\text{For } pl \in DR, \boldsymbol{D}_{GT}(pl) = \sqrt{\left(x_{pl} - xo\right)^2 + (y_{pl} - yo)^2}, o \in GT \tag{2.23a}$$

$$\text{For } pl \in GT, \boldsymbol{D}_{DR}(pl) = \sqrt{\left(x_{pl} - xo\right)^2 + (y_{pl} - yo)^2}, o \in DR \tag{2.23b}$$

For an optimum edge detection evaluation, the metric should calculate both distances of *FPs* and *FNs* as Symmetric Distance Measure (*SDM*) [49, 50] and Relative Distance Error (RDE_L).

- $SDM(GT, DR) = \sqrt[L]{\frac{\sum_{pl\epsilon DR} \boldsymbol{D}_{GT}^{L}(pl) + \sum_{pl\epsilon GT} \boldsymbol{D}_{DR}^{L}(pl)}{|GT \cup DR|}}, L \in R^{+}$ (2.24)
- $RDE_L(GT, DR) = \sqrt[L]{\frac{1}{|DR|}\sum_{pl\epsilon DR} \boldsymbol{D}_{GT}^{L}(pl)} + \sqrt[L]{\frac{1}{|GT|}\sum_{pl\epsilon GT} \boldsymbol{D}_{DR}^{L}(pl)},$ (2.25)

Where L= 1 for [6, 7]
L=2 for [49]

Chapter Three

Design Edge Detection System Based on Fractional Sobel Operator

3.1 Introduction

In the present chapter, the software implementation of the proposed edge detection algorithm will be presented. The MATLAB 2017b package has been used as a tool to implement the video edge detection system software. To validate the proposed edge detection algorithm, a supervised evaluation has been adopted.

3.2 Fractional Order Sobel Filters Derivativeation

Let us consider $f\ (x, y)$ as a digital image, whereas (x, y) denotes spatial coordinates.

To get the five fractional filters of the Sobel operator, the equation (2.3) and (2.4) which represent the Gradient of f *in x and y* directions respectively for classical Sobel are derived by the five Yi_Fei operators based on (G-L) and (R-L) definitions illustrated in the table (3.1) [16]. The first fractional Sobel filter was used in paper [21]. The resulting Gradient will be to power $(1+p)$ as shown in equation (3.1). The Derivation of the fractional Sobel mask in x-direction using operator2 is illustrated in equations (3.2), and the fractional Sobel mask based Yi_Fei-2 in y-direction represents the transpose of its fractional Sobel mask based Yi_Fei-2 in the x-direction.

The Gradient of Fractional Sobel based Yi_Fei-2 in x and y directions respectively are shown in equation(3.3).

$$D^p G_x = G_x^{1+p} \tag{3.1}$$

$$\begin{aligned} g_x^{1+p}(x,y) = \{ & a_{+1}[f(x+2,y+1) + 2f(x+2,y) + f(x+2,y-1) \\ & - f(x,y+1) - 2f(x,y) \; - f(x,y-1)] \\ & + a_0[f(x+1,y+1) + 2f(x+1,y) + f(x+1,y-1) \\ & - f(x-1,y+1) - 2f(x-1,y) \; - f(x-1,y-1)] \\ & + a_{-1}\; [f(x,y+1) + 2f(x,y) + f(x,y-1) - f(x-2,y+1) \\ & - 2f(x-2,y) \; - f(x-2,y-1)] \\ & + a_{-2}[f(x-1,y+1) + 2f(x-1,y) + f(x-1,y-1) \\ & - f(x-3,y+1) - 2f(x-3,y) \; - f(x-3,y-1)] \\ & + \; a_{-3}[f(x-2,y+1) + 2f(x-2,y) + f(x-2,y-1) \\ & - f(x-4,y+1) - 2f(x-4,y) \; - f(x-4,y-1)]\} \end{aligned} \tag{3.2}$$

$$G_x^{1+p} = f \cdot M2_x, G_y^{1+p} = f \cdot M2_y \tag{3.3}$$

The Fractional Sobel mask using Yi_Fei-2 in the x-direction is given below:

$$M2_x = \begin{bmatrix} \left(\frac{p^2}{8} - \frac{3p^3}{16} + \frac{p^4}{16}\right) & \left(\frac{p^2}{4} - \frac{3p^3}{8} + \frac{p^4}{8}\right) & \left(\frac{p^2}{8} - \frac{3p^3}{16} + \frac{p^4}{16}\right) \\ \left(\frac{5p}{4} + \frac{p^2}{8} + \frac{p^3}{4}\right) & \left(\frac{5p}{2} + \frac{p^2}{4} + \frac{p^3}{2}\right) & \left(\frac{5p}{4} + \frac{p^2}{8} + \frac{p^3}{4}\right) \\ \left(\frac{5p}{4} + \frac{p^2}{8} - \frac{p^3}{2}\right) & \left(\frac{5p}{2} + \frac{p^2}{4} - p^3\right) & \left(\frac{5p}{4} + \frac{p^2}{8} - \frac{p^3}{2}\right) \\ \left(-1 - \frac{5p}{4} + \frac{5p^2}{8} + \frac{3p^3}{8}\right) & \left(-2 - \frac{5p}{2} + \frac{5p^2}{4} + \frac{3p^3}{4}\right) & \left(-1 - \frac{5p}{4} + \frac{5p^2}{8} + \frac{3p^3}{8}\right) \\ \left(\frac{6p}{4} + \frac{p^2}{8} - \frac{5p^3}{16} - \frac{p^4}{16}\right) & \left(\frac{6p}{2} + \frac{p^2}{4} - \frac{5p^3}{8} - \frac{p^4}{8}\right) & \left(\frac{6p}{4} + \frac{p^2}{8} - \frac{5p^3}{16} - \frac{p^4}{16}\right) \\ \left(1 - \frac{p^2}{2} - \frac{p^3}{8}\right) & \left(2 - p^2 - \frac{p^3}{4}\right) & \left(1 - \frac{p^2}{2} - \frac{p^3}{8}\right) \\ \left(\frac{p}{4} + \frac{p^2}{8}\right) & \left(\frac{p}{2} + \frac{p^2}{4}\right) & \left(\frac{p}{4} + \frac{p^2}{8}\right) \end{bmatrix}$$

Where $M2_y = [M2_x]^T$

Table (3.1): The five operators of Yi_Fei proposed in [16].

	Yi_FeiPU-1 (G-L)	*Yi_FeiPU-2* (G-L)	*Yi_FeiPU-3* , $p < 0$ (R-L)	*Yi_FeiPU-4* , $p < 0$ (R-L)	*Yi_FeiPU-5*, $0 \leq p < 1$ (R-L)
C_{-1}	$\mathbf{0}$	$\frac{p}{4}+\frac{p^2}{8}$	$\mathbf{0}$	$\mathbf{0}$	$\mathbf{0}$
C_0	$\mathbf{1}$	$1-\frac{p^2}{2}-\frac{p^3}{8}$	$\frac{1}{\Gamma(-p)(-2p)}$	$\frac{1}{\Gamma(-p)(p^2-p)}$	$\frac{1}{\Gamma(2-p)}$
C_1	$-p$	$-\frac{5p}{4}+\frac{p^3}{16}+\frac{p^4}{16}$	$\frac{2^{-p}}{\Gamma(-p)(-2p)}$	$\frac{2^{1-p}-2}{\Gamma(-p)(p^2-p)}$	$\frac{2^{1-p}-2}{\Gamma(2-p)}$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
C_i	$\frac{\Gamma(i-p)}{(i)!\,\Gamma(-p)}$	$\frac{\Gamma(i-p-1)}{(i-1)!\,\Gamma(-p)} \cdot \left(-\frac{p}{4}+\frac{p^2}{8}\right)$	$\frac{i^{-p}-(i-1)^{-p}}{\Gamma(-p)(-2p)}$	$\frac{(1-p)i^{-p}-i^{1-p}+(i-1)^{1-p}}{\Gamma(-p)(p^2-p)}$	$\frac{(1-v)i^{-p}-i^{1-p}(i-1)^{1-p}}{\Gamma(2-p)}$

The other masks using Yi_Fei-3 to Yi_Fei-5 operators are as follow:

$$M3_x = \begin{bmatrix} \left(\frac{-2^{-2p}+3^{-p}}{\alpha}\right) & \left(\frac{-2^{1-2p}+2*3^{-p}}{\alpha}\right) & \left(\frac{-2^{-2p}+3^{-p}}{\alpha}\right) \\ \left(\frac{1-2*3^{-p}+2^{-2p}}{\alpha}\right) & \left(\frac{2-2^2*3^{-p}+2^{1-2p}}{\alpha}\right) & \left(\frac{(1-2*3^{-p}+2^{-2p})}{\alpha}\right) \\ \left(\frac{3^{-p}-2^{-p}-1}{\alpha}\right) & \left(\frac{2-2^2*3^{-p}+2^{1-2p}}{\alpha}\right) & \left(\frac{3^{-p}-2^{-p}-1}{\alpha}\right) \\ \left(\frac{-1+3^{-p}-2^{-p}}{\alpha}\right) & \left(\frac{-2+2*3^{-p}-2^{1-p}}{\alpha}\right) & \left(\frac{3^{-p}-2^{-p}-1}{\alpha}\right) \\ \left(\frac{2^{-p}}{\alpha}\right) & \left(\frac{2^{1-p}}{\alpha}\right) & \left(\frac{2^{-p}}{\alpha}\right) \\ \left(\frac{1}{\alpha}\right) & \left(\frac{2}{\alpha}\right) & \left(\frac{1}{\alpha}\right) \end{bmatrix}$$

Where $\alpha = \Gamma(-p) * (p^2 - p)$, $M3_y = [M3_x]^T$

$$M4_x = \begin{bmatrix} \left(\frac{(1-p)*2^{-2p}-2^{2-2p}+3^{1-p}}{\boldsymbol{\alpha}}\right) & \left(\frac{(1-p)*2^{1-2p}-2^{3-2p}+2*3^{1-p}}{\boldsymbol{\alpha}}\right) & \left(\frac{(1-p)*2^{-2p}-2^{2-2p}+3^{1-p}}{\boldsymbol{\alpha}}\right) \\ \left(\frac{2^{2-p}-2^{2-2p}+(1-p)*2^{-2p}-1}{\boldsymbol{\alpha}}\right) & \left(\frac{2^{2-p}-2^{2-2p}+(1-p)*2^{-2p}-1}{\boldsymbol{\alpha}}\right) & \left(\frac{2^{2-p}-2^{2-2p}+(1-p)*2^{-2p}-1}{\boldsymbol{\alpha}}\right) \\ \left(\frac{3-3*2^{1-p}+3^{-p}}{\boldsymbol{\alpha}}\right) & \left(\frac{6-3*2^{2-p}+2*3^{-p}}{\boldsymbol{\alpha}}\right) & \left(\frac{3-3*2^{1-p}+3^{-p}}{\boldsymbol{\alpha}}\right) \\ \left(\frac{3^{1-p}-2^{2-p}}{\boldsymbol{\alpha}}\right) & \left(\frac{2*3^{1-p}-2^{3-p}}{\boldsymbol{\alpha}}\right) & \left(\frac{3^{1-p}-2^{2-p}}{\boldsymbol{\alpha}}\right) \\ \left(\frac{2^{1-p}-2}{\boldsymbol{\alpha}}\right) & \left(\frac{2^{2-p}-2^2}{\boldsymbol{\alpha}}\right) & \left(\frac{2^{1-p}-2}{\boldsymbol{\alpha}}\right) \\ \left(\frac{\mathbf{1}}{\boldsymbol{\alpha}}\right) & \left(\frac{\mathbf{2}}{\boldsymbol{\alpha}}\right) & \left(\frac{\mathbf{1}}{\boldsymbol{\alpha}}\right) \end{bmatrix}$$

Where $\alpha = 1/\Gamma(2-p),\ M4_y = [M4_x]^T$

$$\mathrm{M5_x} = \begin{bmatrix} \left(\frac{2^{2-2p}-(1-p)*2^{-2p}-3^{1-p}}{\alpha}\right) & \left(\frac{2^{3-2p}-(1-p)*2^{1-2p}-2*3^{1-p}}{\alpha}\right) & \left(\frac{2^{2-2p}-(1-p)*2^{-2p}-3^{1-p}}{\alpha}\right) \\ \left(\frac{-2^{2-p}+3^{1-p}+1}{\alpha}\right) & \left(\frac{-2^{3-p}+2*3^{1-p}+2}{\alpha}\right) & \left(\frac{-2^{2-p}+3^{1-p}+1}{\alpha}\right) \\ \left(\frac{2^{2-p}-2^{2-2p}+(1-p)*2^{-2p}-1}{\alpha}\right) & \left(\frac{2^{3-p}-2^{3-2p}+(2-2p)*2^{-2p}-2}{\alpha}\right) & \left(\frac{2^{2-p}-2^{2-2p}+(1-p)*2^{-2p}-1}{\alpha}\right) \\ \left(\frac{-2^{1-p}+3^{1-p}+3}{\alpha}\right) & \left(\frac{-2^{2-p}+2*3^{1-p}+6}{\alpha}\right) & \left(\frac{-2^{1-p}+3^{1-p}+3}{\alpha}\right) \\ \left(\frac{3^{1-p}-2^{2-p}}{\alpha}\right) & \left(\frac{2*3^{1-p}-2^{3-p}}{\alpha}\right) & \left(\frac{3^{1-p}-2^{2-p}}{\alpha}\right) \\ \left(\frac{2^{1-p}-2}{\alpha}\right) & \left(\frac{2^{2-p}-2^2}{\alpha}\right) & \left(\frac{2^{1-p}-2}{\alpha}\right) \\ \left(\frac{1}{\alpha}\right) & \left(\frac{2}{\alpha}\right) & \left(\frac{1}{\alpha}\right) \end{bmatrix}$$

Where $\alpha = \Gamma(3-p),\ M5_y = [M5_x]^T$

The filtered images by the fractional Sobel filters are thresholding based on equation (3.4). each pixel in the filtered image is considered as an edge if the magnitude of gradient for the pixel is greater than $Thresold$ as shown in equation (3.5).

$$Thresold = Scale \cdot \ \overline{G} \tag{3.4}$$

$$DR = |G| > Threshold \tag{3.5}$$

Where Scale ≥ 1, $\bar{G}$ is the average of$|G|$, and DR is the detector result.

3.3 Software Development

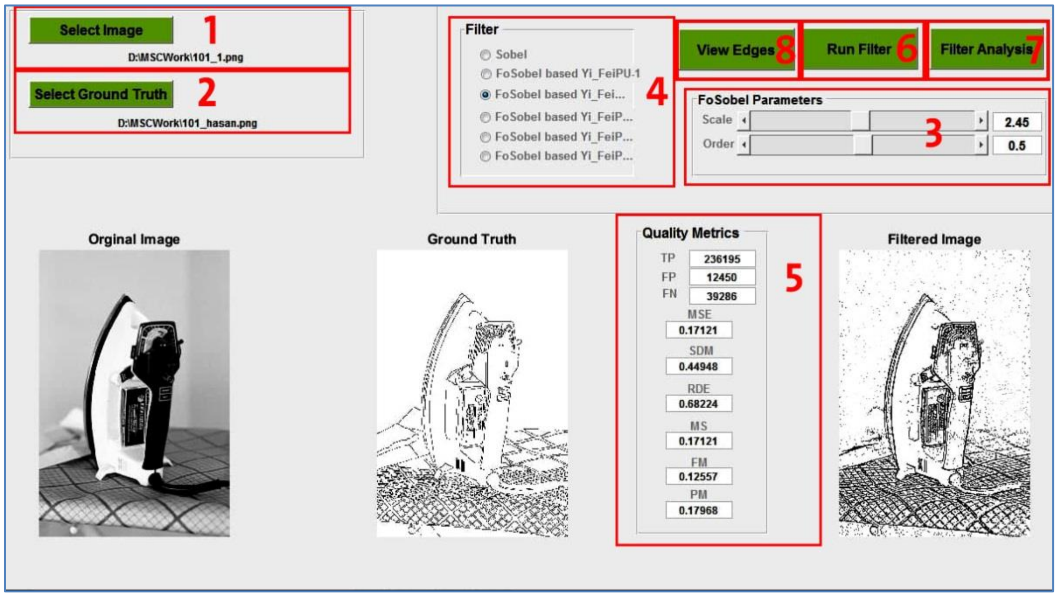
The Graphical User interface (GUI) of the fractional Sobel edge detection system software has been implemented in Matlab, as shown in figure (3.1). The GUI source code is presented in the appendix A. The GUI involves the following items:

1- Select Image: To select the source image by using the windows application interface (API) function browse file.

2- Select Ground Truth: To select its Ground truth image by using API browse file.

3- FoSobel Parameters: To select detector fractional order (p) and *Scale*. It can be manually passed from the *Scale* and order slide bars or edited.

4- Filter: To select the type of detector (classical Sobel, fractional Sobel based Yi_Fei-1, fractional Sobel based Yi_Fei-2, fractional Sobel based Yi_Fei-3, fractional Sobel based Yi_Fei-4, fractional Sobel based Yi_Fei-5).



D:\MSCWork\101_1.png

D:\MSCWork\101_hasan.png

Filter
- ○ Sobel
- ○ FoSobel based Yi_FeiPU-1
- ● FoSobel based Yi_Fei...
- ○ FoSobel based Yi_FeiP...
- ○ FoSobel based Yi_FeiP...
- ○ FoSobel based Yi_FeiP...

FoSobel Parameters

Scale	2.45
Order	0.5

Quality Metrics

TP	236195
FP	12450
FN	39286
MSE	0.17121
SDM	0.44948
RDE	0.68224
MS	0.17121
FM	0.12557
PM	0.17968

Figure (3.1): GUI system of the fractional Sobel edge detection.

5- Quality Metrics: Detectors assessment are implemented based on the following metrics which are presented in equations (2.19) to (2.25):
   a) Mean Square Error.
   b) Symmetric Distance Measure.
   c) Relative Distance Error.
   d) Misclassification Error.
   e) Complemented Performance Measure.
   f) Complemented F Measure.
6- Run Filter: To filter the source image using the selected operator and views metrics score on the Quality Metrics panel.
7- Filer Analysis: To analyze the understudy filters at fractional order ($o \leq p < 1$, in steps of 0.01) based on supervised assessment. All the results are saved in an Excel file.
8- View Edges: To view the result edge maps of the filters at specified parameters (initial order value, step value, final order value) using the Matlab montage built-in function.

3.4 Results Analysis

To validate the best operator; a supervised evaluation on the proposed operators and classical Sobel operator is performed. Test images and their ground truth have been used in the evaluation process. The Ground truth image is an edge map acquired from human judgment or synthetic data. Test images have been filtered using the proposed filters with orders(*p*) from 0 to 1in steps of 0.01. The results of utilized assessments are illustrated in figures (3.2), (3.3), (3.4), (3.5), and (3.6) respectively.

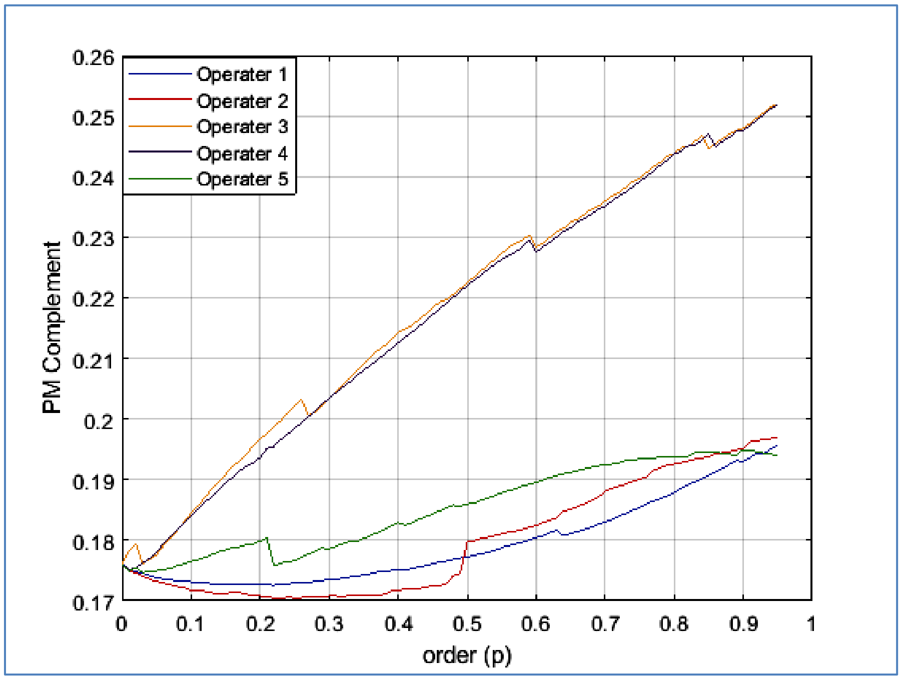


Figure (3.2): Complemented Performance Measure for the five fractional-order Sobel filters.

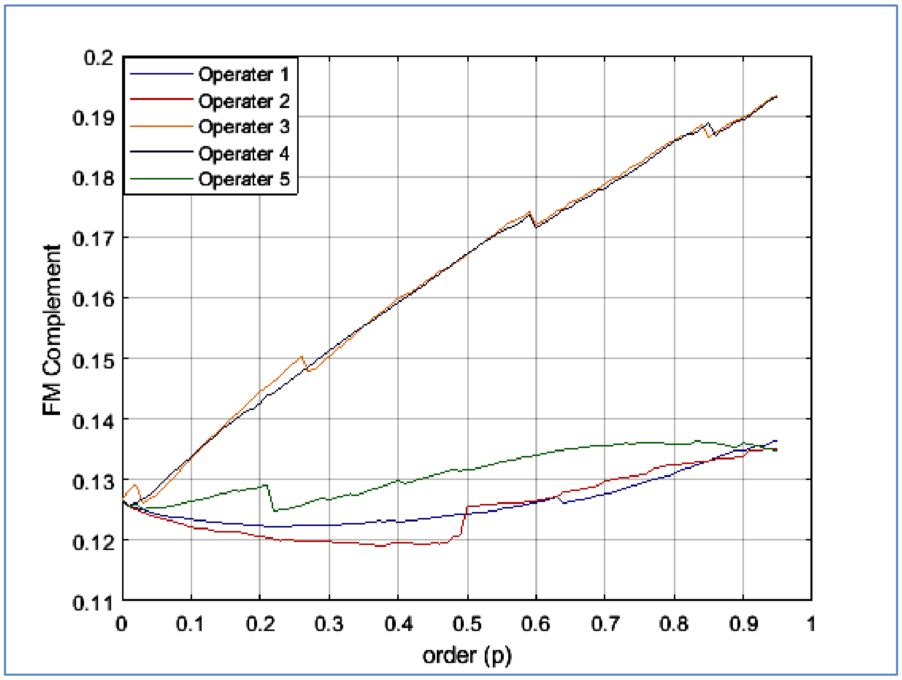


Figure (3.3): Complemented F Measure for the five fractional-order Sobel filters.

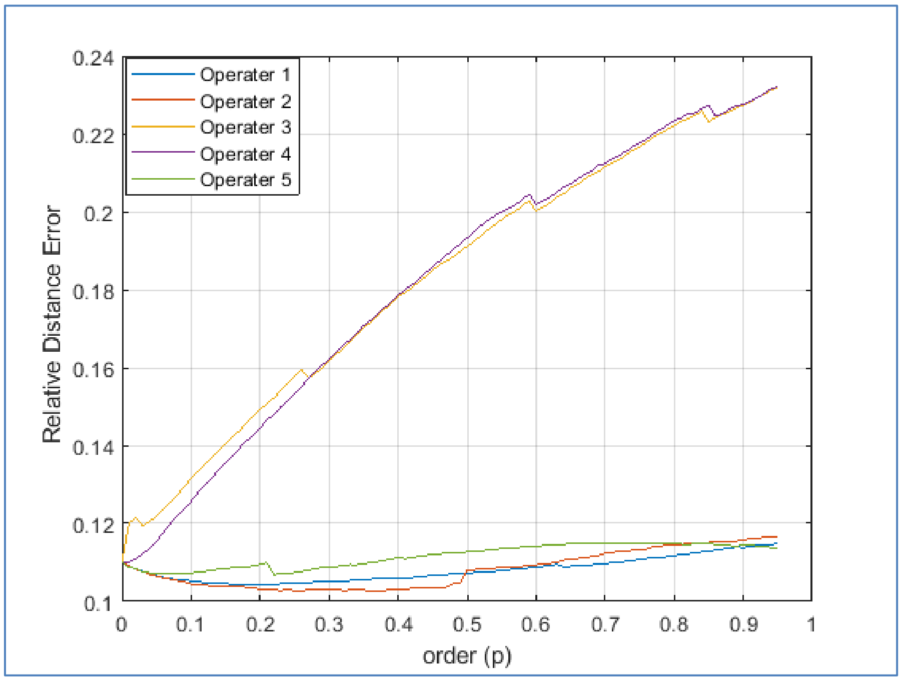


Figure (3.4): Relative Distance Error for the five fractional-order Sobel filters.

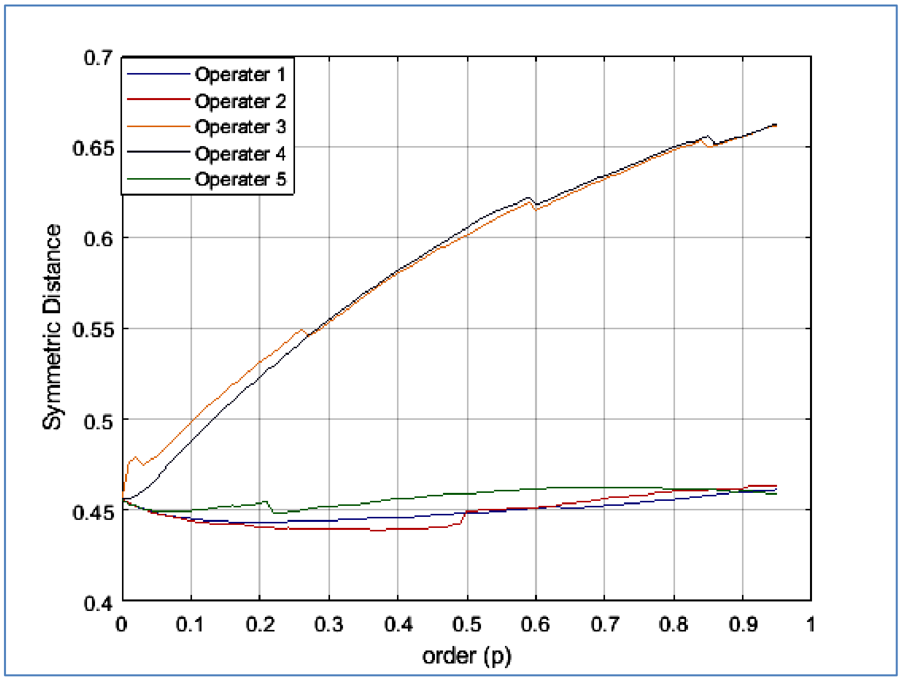


Figure (3.5): Symmetric distance for the five fractional-order Sobel filters.

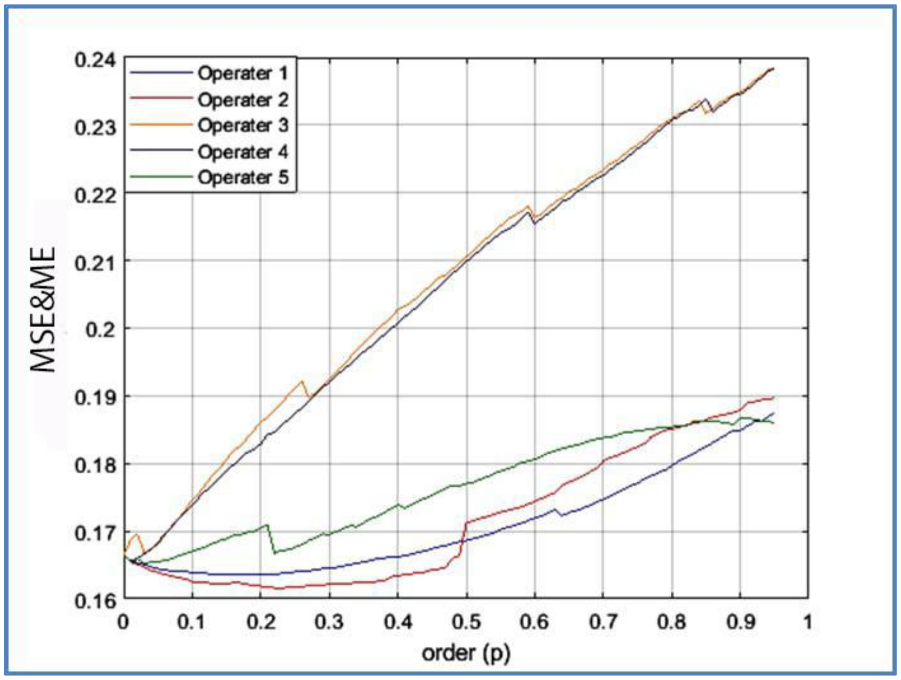


Figure (3.6): MSE and ME for the five fractional-order Sobel filters.

According to the scores of PM, FM, RDE_L, SDM, MSE, and ME the fractional-order Sobel filter based on Yi_Fei-2 has a better performance than that of the classical Sobel filter and the other proposed fractional operators because it qualifies the best scores of evaluators (minimum scores which close to zero) when the order greater than 0.2 and less than 0.4 as demonstrated in table (3.2).

From the scores of *MSE* and *ME,* as shown in figure (3.6), it has been noticed that they have the same values for all the fractional Sobel and classical one at all values of (p) because GT and *DR* are binary images. To justify this: The Mean Square Error as illustrated in equation (2.22), where *MN* is the total pixels of the binary edge map image which equal to:

$MN=TP+TN+FP+FN,$

$\sum_{i=1}^{M}\sum_{j=1}^{N}\left(Gt_{i,j}-Dr_{i,j}\right)^2 = FN+FP$

$$MSE=\frac{FP+FN}{TP+FP+FN+TN} \tag{3.6}$$

The Misclassification Error as shown in equation (2.19):

$ME = 1-\frac{TN+TP}{TP+TN+FN+FP}$

$$ME=\frac{(TP+TN+FN+FP)-(TN+TP)}{TP+TN+FN+FP}=\frac{FP+FN}{TP+FP+FN+TN} \tag{3.7}$$

Equations (3.6) and (3.7) prove that the Mean Square Error and Misclassification Error are equal when used to evaluate a binary edge map. Whereas the scores of *TP*, *FN,* and *Fp* are shown in figures (3.7), (3.8), and (3.9) respectively for all operators, as illustrated the fractional-order Sobel filter based on Yi_Fei-2 has maximum scores of true edge points (*TP*) and minimum scores of missing points (*FN*) which refer to its good performance to detect the true edges in range (0.1<*p<0.47)*. The best edge detector should have minimum scores of false edges (*FP*) but the fractional-order Sobel filter based on Yi_Fei-2 has maximum scores of *FP*, indeed there are numerous edges in the source image that doesn't take in GT and the FP distances of the fractional-order Sobel filter based on Yi_Fei-2 from the actual edge points are the minimum scores comparing to other proposed filters and the classical one as shown in *SDM* and *RDEL* scores. Values of Max.TP and Min.FN, FP of classical Sobel and fractional Sobel operators shown in table (3.3). Figure (3.10) shows the result edge map for FS_Yi_Fei-2 at the order (*p*=0.25).

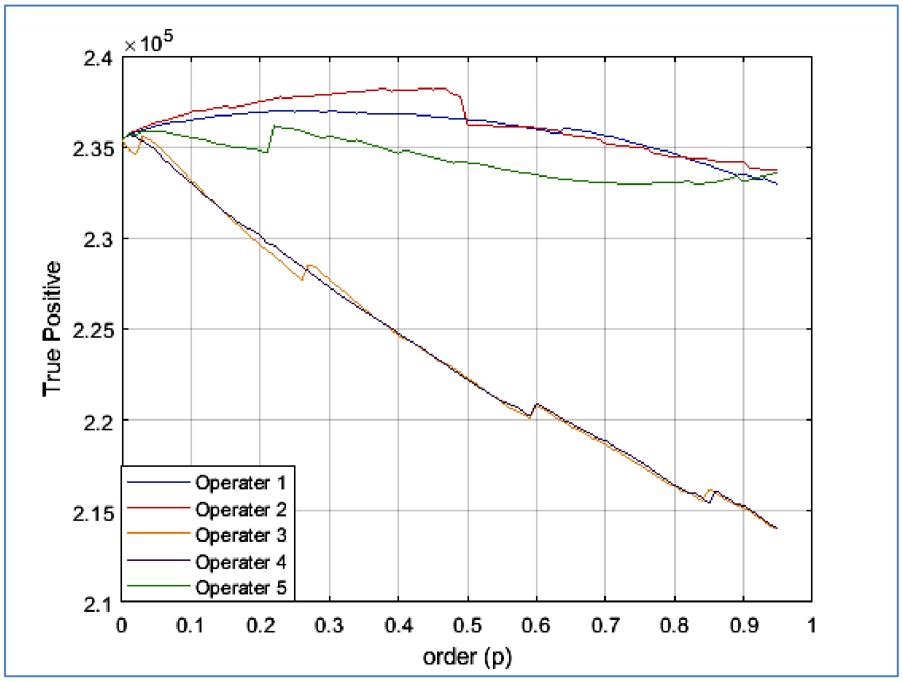


Figure (3.7): TP for the five fractional-order Sobel filters.

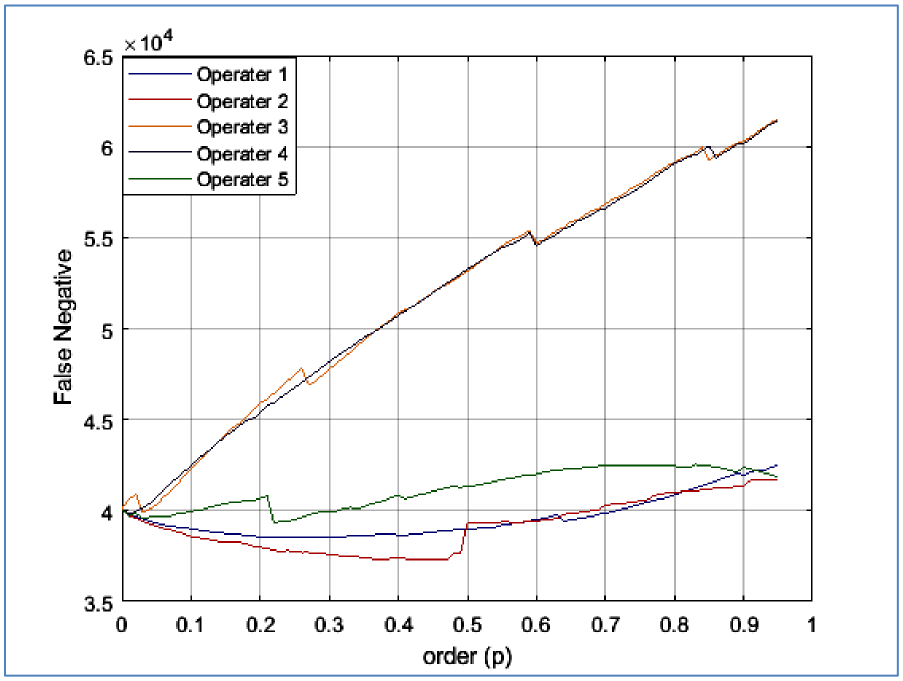


Figure (3.8): FN for the five fractional-order Sobel filters.

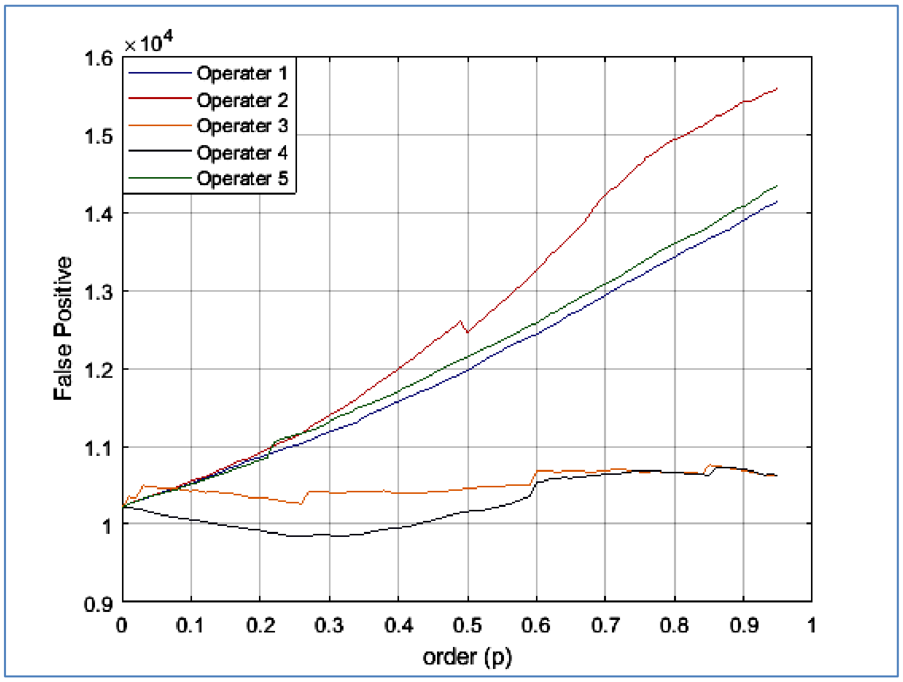


Figure (3.9): FP for the five fractional-order Sobel filters.

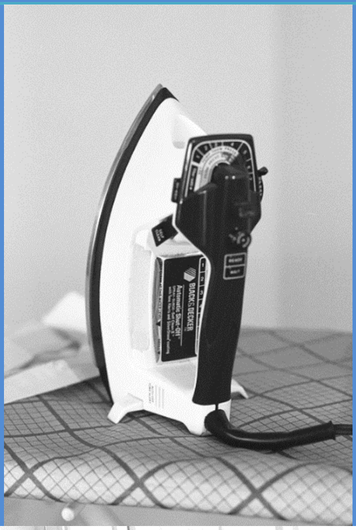
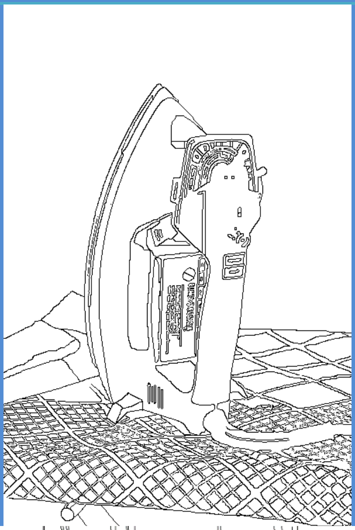
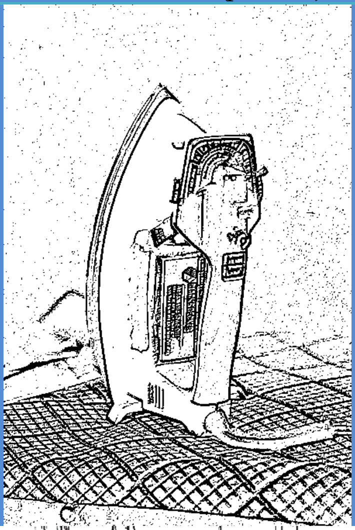
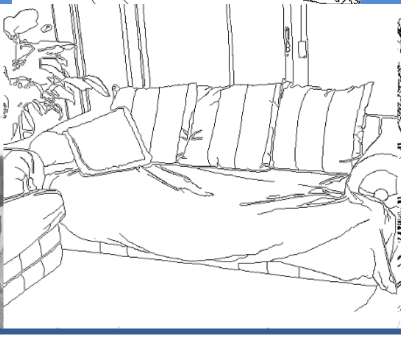
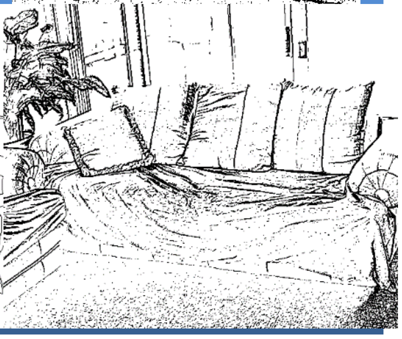
Original	Ground truth	Fractional Sobel filter based Yi_Fei-2 (*p=0.25*)
		
		

Figure (3.10): Edge detection using fractional Sobel filter based Yi_Fei-2.

Important notice on FS_Yi_Fei-2 is that when selecting the ground truth as the original image and comparing the DR with GT, the scores of metrics are close to zero and enhance the existed edges in the original image (*GT*) as illustrated in table (3.4). This enhancement proves the benefits of fractional order derivative among integer derivative in preserving more low-frequency contour features in smooth areas, keep high-frequency marginal features where grayscale changes frequently, and also enhancing medium-frequency texture details. This is a proposed way to validate edge detector performance. Figure (3.11) shows the filtered image by Sobel and

FS_Yi_Fei-2 when GT is selected as the source image, and the scores of metrics at (*p*=0.5) are illustrated in table(4.3).

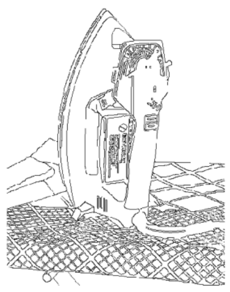
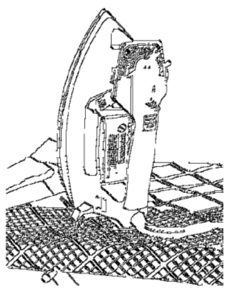
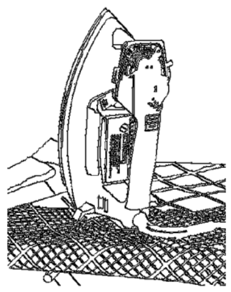
Original as Ground truth	**Sobel filter edge map**	**FS_Yi_Fei-2 edge map (*p=0.5)***
		

Figure (3.11): Fractional Sobel filter based Yi_Fei-2 and sobel operator edge maps when original is the ground truth.

Table (3.2):Minimum Scores of Fractional Sobel operator Yi_Fei-2.

Measure	**Value**	**Order(*p*)**
MSE	0.1614	0.23
SDM	0.4391	0.38
RDE	0.6674	0.37
ME	0.1614	0.23
PM	0.1191	0.38
FM	0.1702	0.23

Table (3.3): Values of Max.TP and Min.FN, FP of classical Sobel and Fractional Sobel operators.

Name	**TP**	**FN**	**FP**
Sobel	235376	40105	10208
FS_Yi_Fei-1	236965	38516	10323
FS_Yi_Fei-2	238197	37284	10261
FS_Yi_Fei-3	235590	39891	10256
FS_Yi_Fei-4	235671	39810	9848
FS_Yi_Fei-5	236162	39319	10262

Table (3.4): Scores of Fractional Sobel operator Yi_Fei-2 and Sobel operator when select GT as the original image.

Measure	**FS_Yi_Fei-2 Scores(p=0.5)**	**Sobel operator**
TP	245493	242995
FP	1092	14060
FN	29988	32486
MSE	0.10285	0.15404
SDM	0.34915	0.46988
RDE	0.43438	0.6051
ME	0.10285	0.15404
PM	0.11238	0.16076
FM	0.089761	0.10596

Chapter Four

FPGA Implementation of Video Edge Detection

4.1 Introduction

The implementation of the proposed fractional Sobel filter based on the Yi_Fei-2 operator will be presented in the present chapter. Xilinx Artix-7 NEXYS 4 kit has been used as a platform for the implementation. Xilinx Vivado 2018.2 and MATLAB 2017b are the developing environment of the implemented system. The hardware system has been designed through using a high-level Simulink graphical environment of Xilinx system generator (XSG) that provides graphical modules which overcome the Hardware Description Language (HDL) coding. The Simulink helps in abstracting the design using XSG blocks and subsystems which specifically reduces the hardware implementation time. Also, Simulink provides FPGA in the loop (FIL) simulation and verification. This approach makes the hardware verification and implementation easier in contrast with HDL based approach. FIL methodology compared to other methodologies, presents a more cost-effective solution [26, 28].

4.2 Hardware in the Loop

One of the hardware Co-simulation methods is the hardware in the loop (HIL). In HIL methodology, it is possible to integrate a design running in an FPGA directly into a simulator. Figure (4.1) illustrates the hardware design simulated in Simulink and processed in real FPGA hardware. HIL automatically creates a bitstream of associated hardware block. HIL Co-simulation is often significantly faster than classical simulation while testing the hardware's functional correctness. HIL supports Ethernet and JTAG communication between the Simulink and a hardware platform.

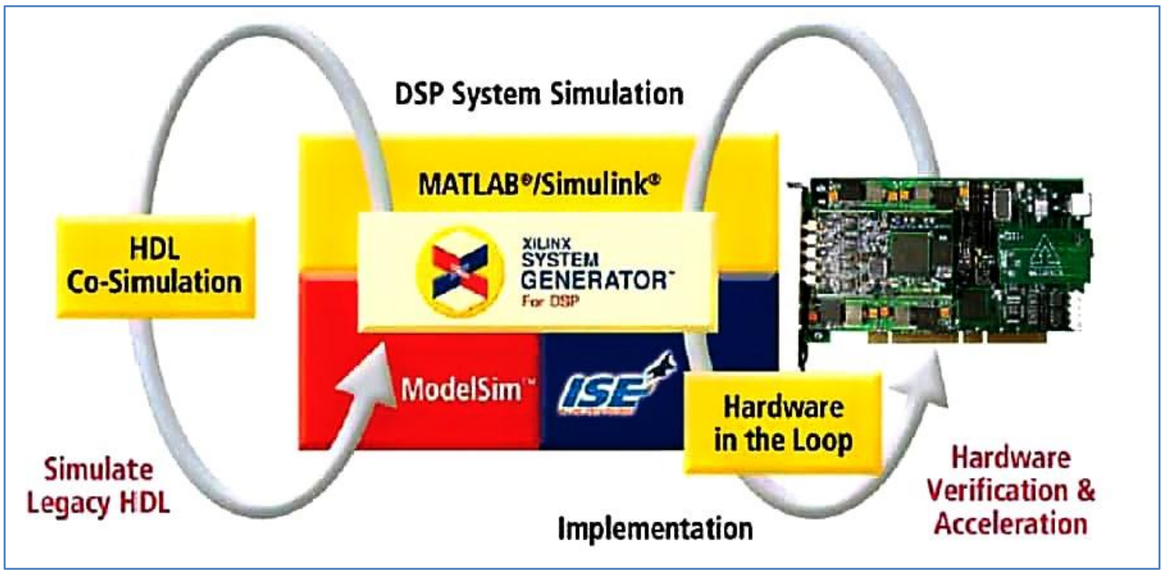


Figure (4.1): HIL Co-simulation [26].

HIL methodology is a mixture of software and hardware systems that gets the benefits of both. HIL approach works in real-time; though, the surrounding components are simulated in the software environment. This approach enables device flexibility with real-world precision and hardware speed of execution [26,29].

4.3 Proposed Edge Detection System

The developing platform of Matlab Simulink encapsulates system design, system modeling, system simulation, system code generation, and system implementation. Figure (4.2) displays the proposed edge detection system using the fractional Sobel filter-based Yi_Fei-2 operator. The fractional Sobel filter-based Yi_Fei-2 operator model is to be implemented in Artix-7 NEXYS 4 board.

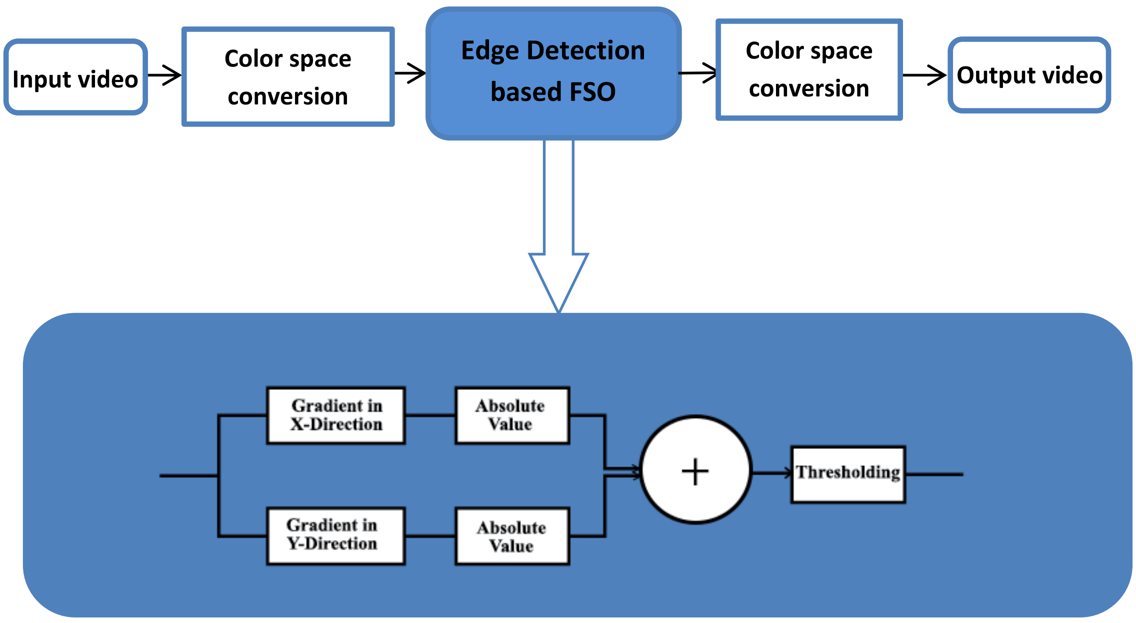


Figure 4.2: Edge Detection System.

4.4 Implementation of the Proposed System

The Matlab vision HDL toolbox provides Simulink blocks that support image and video processing in the HDL environment as shown in figure (4.3). The video source block provides a frame stream from a video file. Frame to pixel block transforms the input frame stream to pixels stream with control signals (vStart, vEnd, hStart, hEnd, and valid data), the frame size can be configured by using the block properties. The HDL algorithm block contains a fractional Sobel filter-based Yi_Fei-2 operator. The pixel-to-frame block transforms the incoming pixels stream to frame stream. The video viewer block displays the incoming frames.

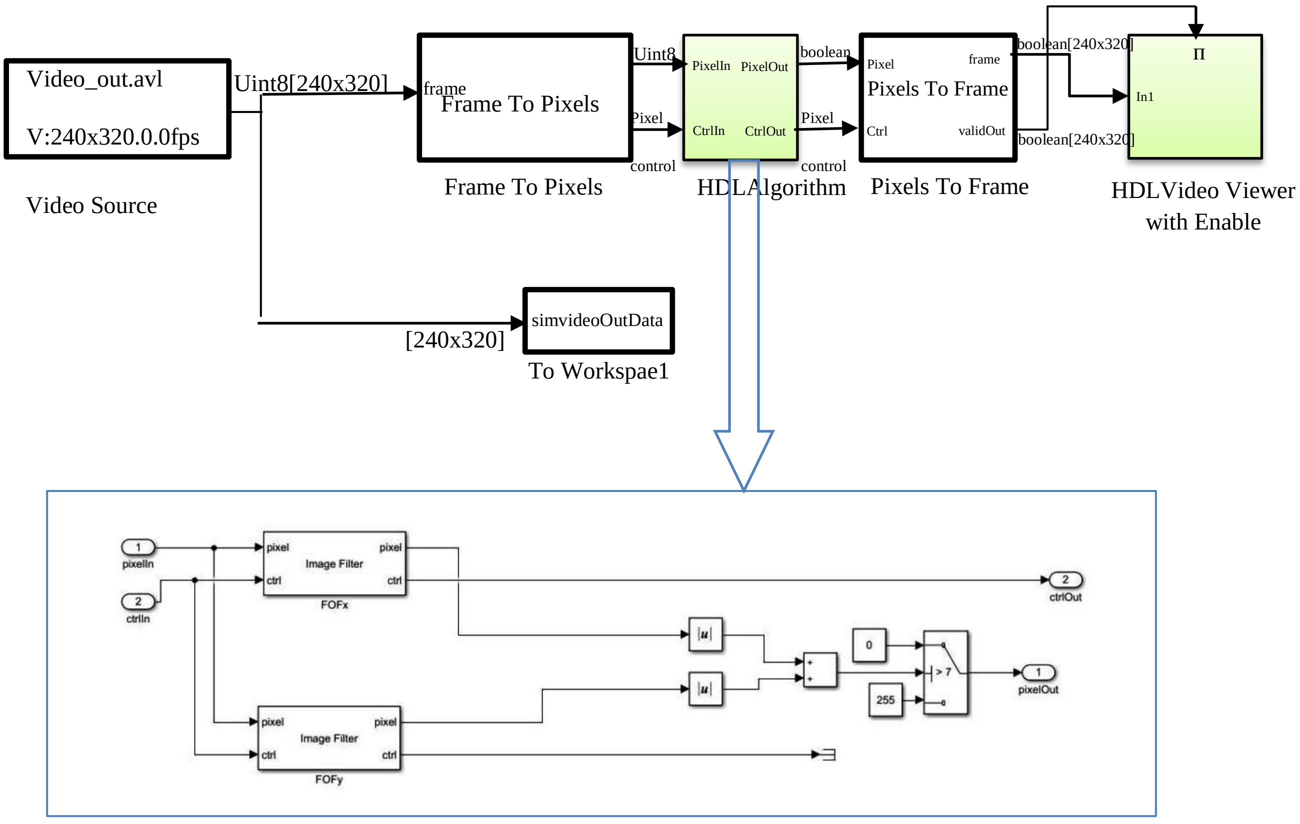


Figure (4.3): The equivalent software implementation of the HDL algorithm.

4.4.1 Hardware Compilation and Co-simulation

Before hardware Co-simulation, the HDL algorithm block must be compiled to generate the HDL code of the block using Matlab XSG. Then using FPGA in the loop wizard (filWizard) to generate the bitstream of the implemented hardware for the specified Xilinx board (Artix-7 NEXYS 4). The filWizard generates hardware Co-simulation block, then invokes Xilinx Vivado 2018.2 to generate the bitstream of the Xilinx chipset according to the configuration constraints of the board. The hardware Co-simulation block is loaded with the bitstream file. figure (4.4) shows the generated hardware Co-simulation block (Edge Detection H/W).

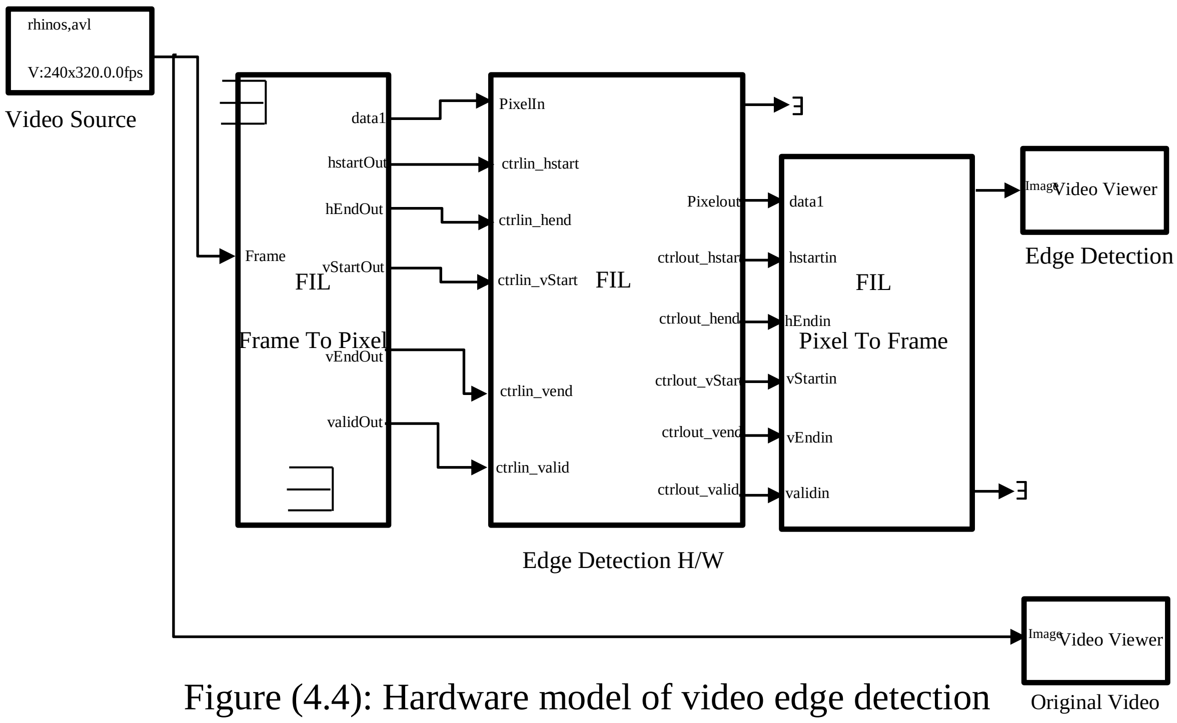


Figure (4.4): Hardware model of video edge detection

4.4.2 Hardware Implementation Reports

The summary of the utilization report for the implemented hardware on Artix-7 NEXYS 4 is illustrated in table (4.1). Figure (4.5) demonstrate the utilization of the Artix-7 NEXYS 4 chipset.

Table (4.1): Hardware utilization of video edge detection.

Resources	**Available**	**Used**	**Utilization**
BUFG	32	1	3.13
BRAM	135	15	11.11
FF	126800	2227	1.76
LUTRAM	19000	166	0.87
LUT	63400	1855	2.93

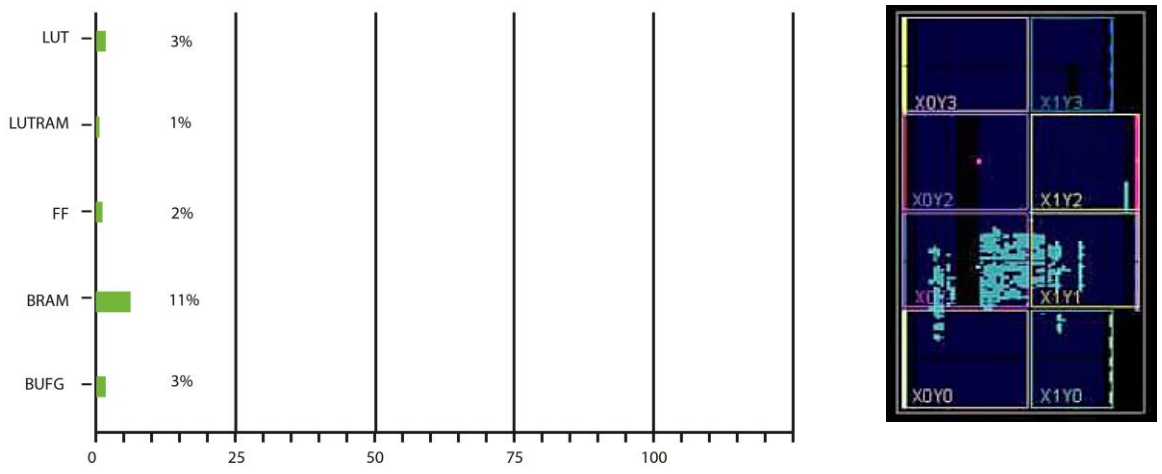


Figure (4.5): Hardware Utilization of video edge detection for Artix-7 NEXYS 4

4.4.3 Results

The FIL Co-simulation is composed of the personal computer, Artix-7 NEXYS 4, and Matlab 2017b as demonstrated in figure (4.6). The video frame is sent via JTAG port to Artix-7 NEXYS 4 board. The frame is processed in the FPGA chipset of the board and then sent back to Simulink. Table (4.2) shows samples of frames for the Rhino video file. Figure (4.7) shows the root mean square error between the simulation results and H/W Co-simulation.

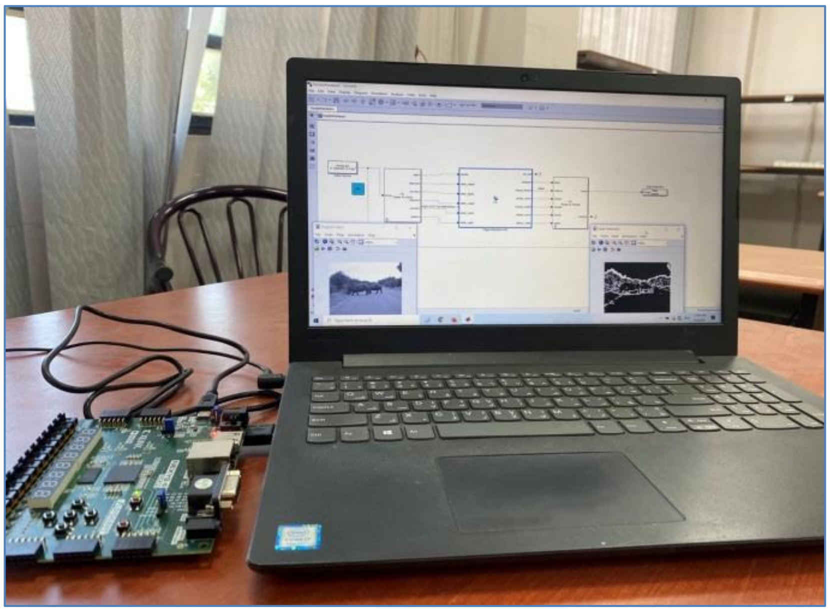
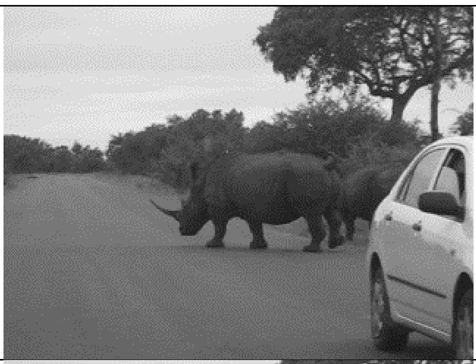
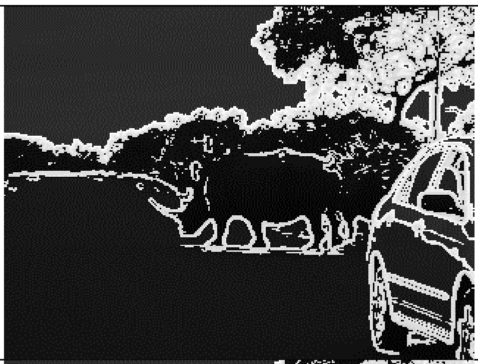
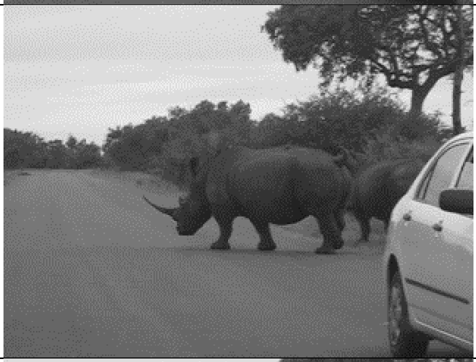
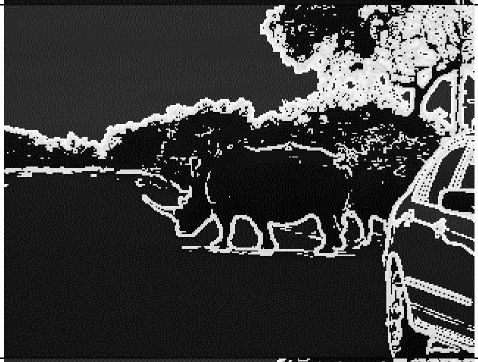
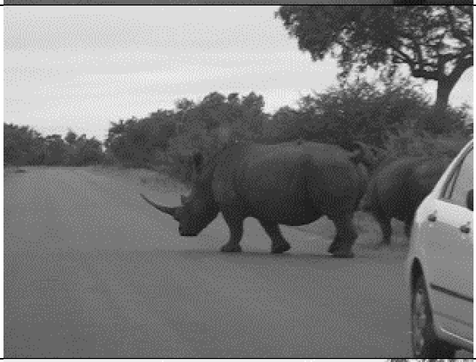
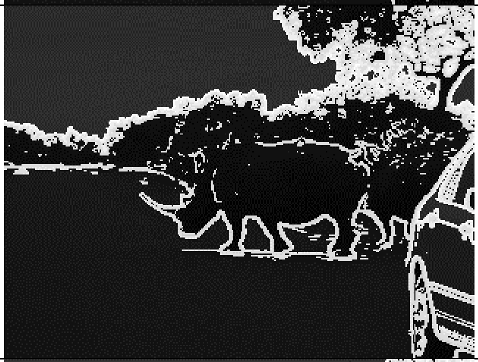
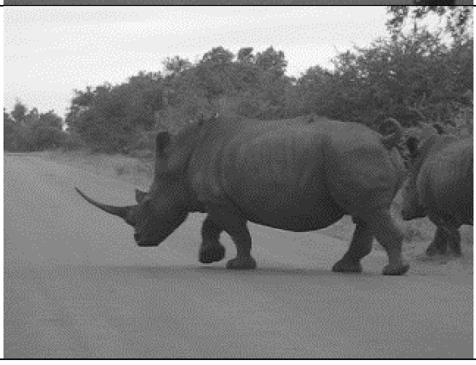
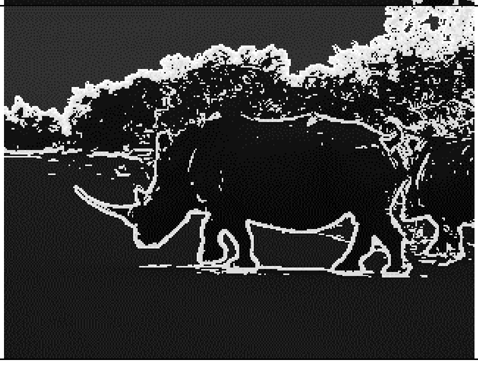
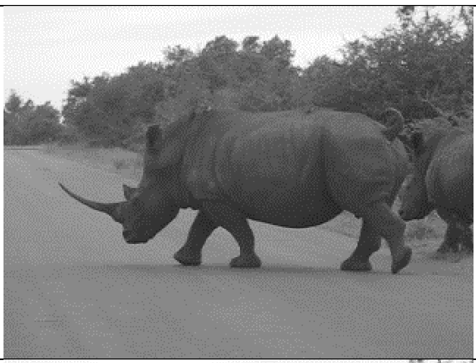
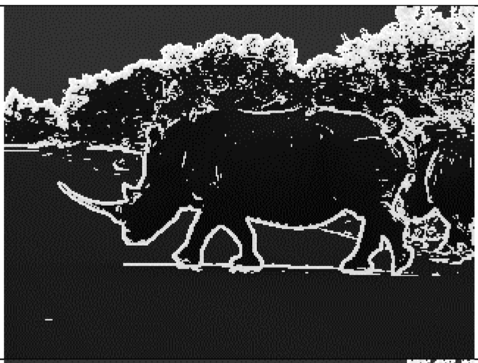
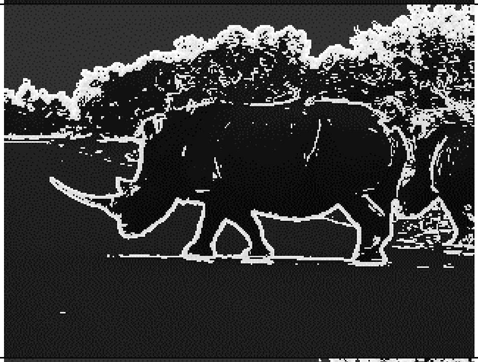
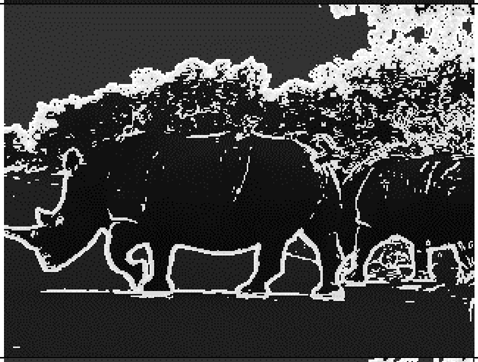
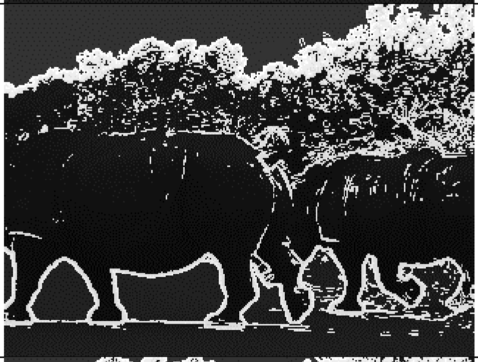
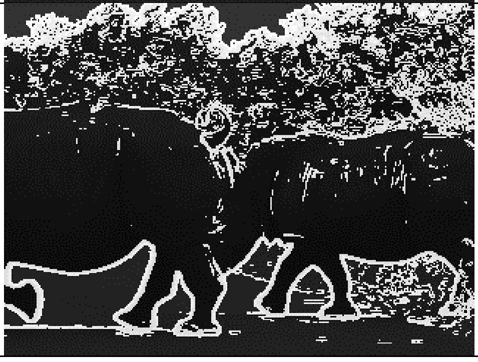
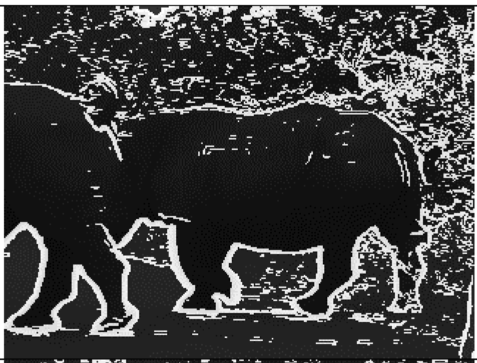
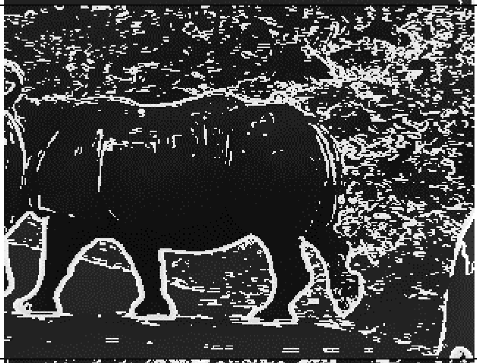
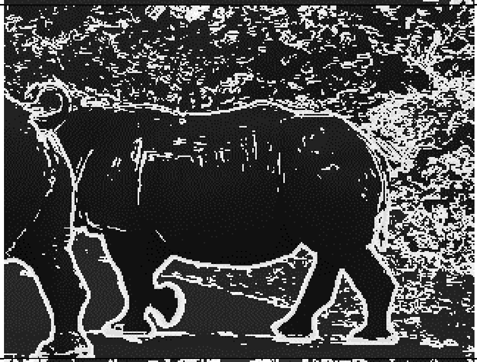
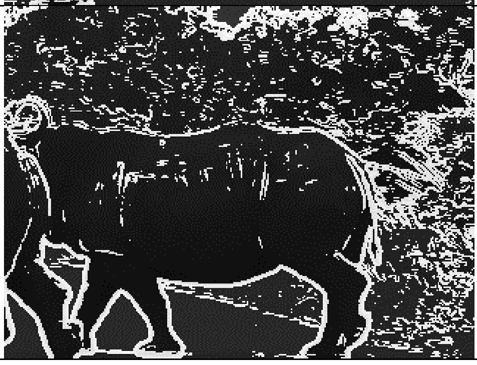


Figure (4.6): System Co-simulation.

Table (4.2): Samples of Co-simulation frames.

Frame No.	Input Frame	Output Frame
1		
2		
3		
9		

11		
14		
20		
25		
30		

35		
40		
45		
50		

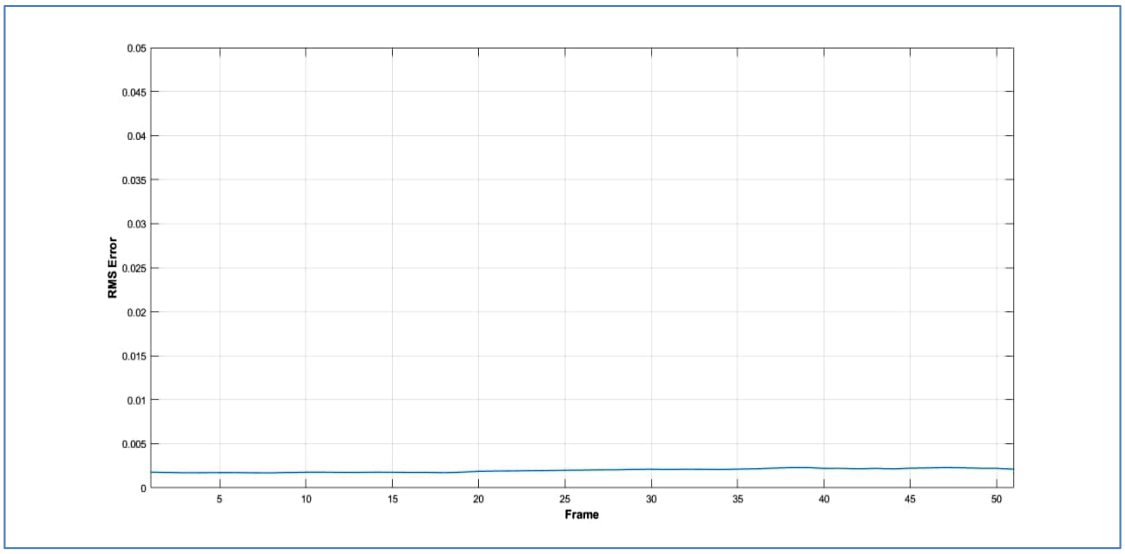


Figure (4.7): Root mean square error between hardware processing and simulation.

Chapter Five

Conclusions and Suggestion for Future Works

5.1 Conclusions

Video edge detection is considered as one of the vital research works and an important key step towards the realization of image features, reduces the amount of data that needs to be processed by extracting only significant information from an image. In the present thesis, the edge detection based on fractional Sobel filter has been designed and evaluated to verify its reliability and then implemented on FPGA by using FIL approach based on Artix7 NEXYS 4 kit. The following points have been concluded:

- The fractional-order Sobel masks based on Yi_Fei-2, Yi_Fei-3, Yi_Fei-4, and Yi_Fei-5 have been designed. Results based on supervised assessments showed that FS_Yi_Fei-2 is the best performance among the other proposed fractional and classical Sobel operators at $(0.2 < p < 0.4)$.
- The fractional-order derivative is better than integer-order in accuracy and flexibility by allowing steerable order which can be increased or decreased to get a different edge map.
- The results showed that the metrics, Mean Square Error, Symmetric Distance Measure, Relative Distance Error, Misclassification Error, Complemented Performance Measure, and Complemented F Measure behave in the same manner except the “true positive”, “false negative”, and “false positive”.
- The scores of Mean Square Error and Misclassification Error are equal if the edge map result and ground truth were binary images (i.e. zero and one image).

- The complement F measure is the most stable among the other error measure as concluded in [10] because it doesn’t take the “true negative” in its calculation because it represents the dominant edge map and reflects an unfair judgment on a detector.
- The FS_Yi_Fei-2 has maximum “true positive” and minimum “false negative” at the same order which indicates its ability to detect true edge more than other proposed operators and classical Sobel operator.
- Many assessments are available but their validity depends on the measurement itself and ground truth image. Thus an incomplete ground truth punishes a detector detecting true pixels as edge or Honors bad one.
- The FPGA in the loop approach has been utilized for the FPGA hardware implementation of the proposed operators. The root mean square error between hardware implementation and software simulation is 0.002.

5.2 Future Works

Some suggestions for further research:

- The improvement of the proposed fractional Sobel filter by using the Hysteresis thresholding which can decrease the number of superiors points “false positive” and test immunity to noise.
- The use of the proposed fractional operator for video segmentation.
- The use of the proposed fractional operator for medical applications.

Reference

[1] R. C. Gonzalez and R. E. Woods, “*Digital image processing,*” Prentice hall Upper Saddle River, Second edition, NJ, 2002.

[2] Pratt, William K, “*Introduction to digital image processing,*” CRC press, 2013.

[3] Claudia I. Gonzalez, Patricia Melin, Juan R. Castro and Oscar Castillo, “*Edge Detection Methods Based on Generalized Type-2 Fuzzy Logic,*” Springer International Publishing, pp. 89, 2017.

[4] D. Tian, D. Li, and Y. Zhang, “Medical image segmentation based on fractional-order derivative,” in *2015 Asia-Pacific Energy Equipment Engineering Research Conference*, pp. 453–456, 2015.

[5] Q. Yang, D. Chen, T. Zhao, and Y. Chen, “Fractional calculus in image processing: a review,” *Fract. Calc. Appl. Anal.*, vol. 19, no. 5, pp. 1222–1249, 2016.

[6] B. Magnier, “An objective evaluation of edge detection methods based on oriented half kernels,” *ICISP*, pp. 80-89, 2018.

[7] H. Abdulrahman, B. Magnier and P. Montesinos, “A new objective supervised edge detection assessment using hysteresis thresholds,” *International Workshop on Brain-Inspired Computer Vision held as part of ICIAP*, pp. 3-14, 2017.

[8] B. Magnier, “Edge detection: a review of dissimilarity evaluations and a proposed normalized measure,” *Multimedia Tools and Applications* 77, no. 8, pp. 9489-9533, 2018.

[9] B. Magnier, H. Abdulrahman, and P. Montesinos, “A review of supervised edge detection evaluation methods and an objective comparison of filtering gradient computations using hysteresis thresholds,” *J. Imaging*, vol. 4, no. 6, p. 74, 2018.

[10] H. Abdulrahman, B. Magnier, and P. Montesinos, “From contours to ground truth: How to evaluate edge detectors by filtering,” 2017.

[11] A. M. Khidhir and N. Y. Abdullah, “FPGA based edge detection using modified sobel filter,” *Int. J. Res. Dev. Eng.*, vol. 2, no. 1, pp. 22–32, 2013.

[12] Poonam Dhankhar, Neha Sahu, “A review and research of edge detection techniques for image segmentation,” *IJCSMC,* Vol. 2, pp.86-92, 2013.

[13] Ş. Öztürk and B. Akdemir, “Comparison of edge detection algorithms for texture analysis on glass production,” *Procedia-Social Behav. Sci.*, vol. 195, pp. 2675–2682, 2015.

[14] L. Bhargav, D. Nagaraj, and P. Kumar Pareek, “Dynamic resolution of image edge detection technique among Sobel, Log, and Canny algorithms,” *International Journal of Scientific Research Engineering & Technology (IJSRET),* vol. 5.4, pp. 206-210, 2016.

[15] K. B. Krishnan, S. P. Ranga, and N. Guptha, “A survey on different edge detection techniques for image segmentation,” *Indian J. Sci. Technol.*, vol. 10, no. 4, pp. 1–8, 2017.

[16] Y. Pu, J. Zhou and X. Yuan, “Fractional differential mask: a fractional differential-based approach for multiscale texture enhancement,” *IEEE Trans. image Process.*, vol. 19, no. 2, pp. 491–511, 2009.

[17] S. M. Ismail, A. G. Radwan, A. H. Madian, and M. F. Abu-ElYazeed, “Comparative study of fractional filters for Alzheimer disease detection on MRI images,” *2016 39th International Conference on Telecommunications and Signal Processing (TSP),* pp. 720–723, 2016.

[18] W. S. ElAraby, A. H. Madian, M. A. Ashour, “Fractional edge detection based on genetic algorithm,” *2017 29th International Conference on Microelectronics (ICM)*, pp. 1–4, 2017.

[19] R. John and N. Kunju, “Optimization of grunwald-letnikov’s (gl) based fractional filter used for image enhancement,” in *2018 Second International Conference on Inventive Communication and Computational Technologies (ICICCT)*, pp. 612–614, 2018.

[20] D. Li, C. Zhao, M. Jiang, Y. Huang, and Y. Li, “Fractional order edge detection method,” in *2019 IEEE 11th International Conference on Communication Software and Networks (ICCSN)*, pp. 529–534, 2019.

[21] C. Yaacoub and R. A. Z. Daou, “Fractional Order Sobel Edge Detector,” in *2019 Ninth International Conference on Image Processing Theory, Tools and Applications (IPTA)*, pp. 1–5, 2019.

[22] M. Mekideche and Y. Ferdi, “Edge detection optimization using fractional order calculus,” *Int. Arab J. Inf. Technol.*, vol. 16, no. 5, pp. 827–832, 2019.

[23] J. E. Lavín-Delgado, J. E. Solís-Pérez, J. F. Gómez-Aguilar, and R. F. Escobar-Jiménez, “A new fractional-order mask for image edge detection based on Caputo–Fabrizio fractional-order derivative without singular kernel,” *Circuits, Syst. Signal Process.*, vol. 39, no. 3, pp. 1419–1448, 2020.

[24] M. Hacini, F. Hachouf, and A. Charef, “A bi-directional fractional-order derivative mask for image processing applications,” *IET Image Process.*, vol. 14, no. 11, pp. 2512–2524, 2020.

[25] D. Sadykova and A. P. James, “Quality assessment metrics for edge detection and edge-aware filtering: A tutorial review,” *International*

Conference on Advances in Computing, Communications and Informatics (ICACCI), pp. 2366–2369, 2017.

[26] 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*, pp. 852–855, 2012.

[27] P. K. Dash, S. Pujari, and S. Nayak, "Implementation of edge detection using FPGA & model based approach," *International Conference on Information Communication and Embedded Systems (ICICES2014)*, pp. 1–6, 2014.

[28] J. C. T. Hai, O. C. Pun and T. W. Haw, "Accelerating video and image processing design for FPGA using HDL coder and simulink,"*2015 IEEE Conference on Sustainable Utilization And Development In Engineering and Technology (CSUDET)*, pp. 1-5, 2015.

[29] M. Alareqi, R. Elgouri, K. Mateur, A. Zemmouri, A. Mezouari and L. Hlou, "Optimization of high-level design edge detect filter for video processing system on FPGA," 2017 Intelligent Systems and Computer Vision (ISCV), Fez, Morocco, pp. 1-8, 2017.

[30] O. H. Moustafa and S. M. Ismail, "FPGA-based Floating Point Fractional Order Image Edge Detection," in *2019 15th International Computer Engineering Conference (ICENCO)*, pp. 91–94, 2019.

[31] A. Nandal *et al.*, "Image edge detection using fractional calculus with feature and contrast enhancement," *Circuits, Syst. Signal Process.*, vol. 37, no. 9, pp. 3946–3972, 2018.

[32] D. Marr and E. Hildreth, "Theory of edge detection," *Proc. R. Soc. London. Ser. B. Biol. Sci.*, vol. 207, no. 1167, pp. 187–217, 1980.

[33] Z. Othman, H. Haron, M. R. A. Kadir, and M. Rafiq, "Comparison of Canny and Sobel edge detection in MRI images," *Comput. Sci. Biomech. Tissue Eng. Group, Inf. Syst.*, pp. 133–136, 2009.

[34] G. T. Shrivakshan and C. Chandrasekar, "A comparison of various edge detection techniques used in image processing," *IJCSI International Journal of Computer Science Issues*, vol. 5, no. 9, pp. 272-276, 2012.

[35] S. Israni and S. Jain, "Edge detection of license plate using Sobel operator," 2016 International Conference on Electrical, Electronics, and Optimization Techniques (ICEEOT), Chennai, India, pp. 3561-3563, 2016.

[36] Mehra, Rajesh, and Rupinder Verma, "Area efficient fpga implementation of sobel edge detector for image processing applications," *International Journal of Computer Applications* vol. 56,no.16 ,2012.

[37] S. Ghodrati, M. Mohseni, and S. G. Kandi, "Application of image edge detection methods for precise estimation of the standard surface roughness parameters: polypropylene/ethylene-propylene-diene-monomer blend as a case study," *Measurement*, vol. 138, pp. 80–90, 2019.

[38] G.M.H.Amer and A.M.Abushaala, "Edge detection methods," *2015 2nd World Symposium on Web Applications and Networking (WSWAN),* Sousse, Tunisia, pp. 1-7, 2015.

[39] Z. Hussain and D. Agarwal, "A comparative analysis of edge detection techniques used in flame image processing," *Int. J. Adv. Res. Sci. Eng. IJARSE,* no. 4, 2015.

[40] J. Chen, C. Huang, Y. Du and C. Lin, “Combining fractional-order edge detection and chaos synchronisation classifier for fingerprint identification,” *IET Image Process.*, vol. 8, no. 6, pp. 354-362, 2014.

[41] Ding, Lijun, and Ardeshir Goshtasby, “On the Canny edge detector,” *Pattern Recognition* , vol.34, no.3, pp.721-725, 2001.

[42] R. Jain, R. Kasturi, and B. G. Schunck, “*Machine vision*,” vol. 5. McGraw-hill New York, 1995.

[43] S. Kumar, R. Saxena, and K. Singh, “Fractional Fourier transform and fractional-order calculus-based image edge detection,” *Circuits, Syst. Signal Process.*, vol. 36, no. 4, pp. 1493–1513, 2017.

[44] B. G. Yazgaç and M. Kırcı, “Fractional order calculus based fruit detection,” in *2019 8th International Conference on Agro-Geoinformatics (Agro-Geoinformatics)*, pp. 1–4, 2019.

[45] Sezgin, Mehmet, and Bülent Sankur, “Survey over image thresholding techniques and quantitative performance evaluation,”*Journal of Electronic imaging*, vol.13,no.1, pp. 146-165 , 2004.

[46] C. Grigorescu, N. Petkov and M. A. Westenberg, “Contour detection based on nonclassical receptive field inhibition,” *IEEE Transactions on Image Processing*, vol. 12, no. 7, pp. 729-739, 2003.

[47] D. R. Martin, C. C. Fowlkes and J. Malik, “Learning to detect natural image boundaries using local brightness, color, and texture cues,”*IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 26, no. 5, pp. 530-549, 2004.

[48] S. Yang-Mao, Y. Chan and Y. Chu, “Edge Enhancement Nucleus and Cytoplast Contour Detector of Cervical Smear Images,” *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 38, no. 2, pp. 353-366, 2008.

[49] M. -P. Dubuisson and A. K. Jain, “A modified Hausdorff distance for object matching,” *Proceedings of 12th International Conference on Pattern Recognition, Jerusalem, Israel*, pp. 566-568 vol.1, 1994.

[50] C. Lopez-Molina, B. De Baets and H. Bustince, “Quantitative error measures for edge detection,” *Pattern Recognition*, vol. 46, no. 4, pp. 1125-1139, 2013.

[51] S. Singh and R. Singh, “Comparison of various edge detection techniques,” *2015 2nd International Conference on Computing for Sustainable Global Development (INDIACom), IEEE*, pp. 393-396, 2015.

Appendix A

Source Code

```
function varargout = gui_edge(varargin)
% GUI_EDGE MATLAB code for gui_edge.fig
%
% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @gui_edge_OpeningFcn, ...
                   'gui_OutputFcn',  @gui_edge_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT


% --- Executes just before gui_edge is made visible.
function gui_edge_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% Choose default command line output for gui_edge
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);
clc;
%================================================
%Initialization
% ===============================================
global Im1;
global X;
global Im2;
global Xgt;
global F_Index;
%================================================
Im1=0;
Im2=0;
F_Index=2;

%================================================
set(handles.radiobutton3,'Value',1);
set(handles.axes1,'XColor', 'none','YColor','none');
set(handles.axes2,'XColor', 'none','YColor','none');
set(handles.axes3,'XColor', 'none','YColor','none');

title(handles.axes1,sprintf('Orginal Image'),'fontsize',12);
title(handles.axes2,sprintf('Ground Truth'),'fontsize',12);
title(handles.axes3,sprintf('Filtered Image'),'fontsize',12);

%==============================================================
b=get(handles.slider1,'Value');
set(handles.edit1,'String',num2str(b));
%==============================================================
b=get(handles.slider2,'Value');
set(handles.edit2,'String',num2str(b));

% --- Outputs from this function are returned to the command line.
function varargout = gui_edge_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;


% --- Executes on button press in pushbutton1.
function pushbutton1_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton1 (see GCBO) select image
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global X;
global Im1;
[file,folder]= uigetfile( ...
       {'*.png'; ...
        '*.jpg'; ...
        '*.tiff'; ...
        '*.gif'}, ...
        'Enter image');

filename1 =fullfile(folder,file);
set(handles.text2,'String', filename1);
if length(filename1)>4
    X=imread(filename1);
    imshow(X,'Parent',handles.axes1);
    Im1=1;
end
title(handles.axes1,sprintf('Orginal Image'),'fontsize',12);

% --- Executes on button press in pushbutton2.
function pushbutton2_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton2 (see GCBO) Select GT
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global Xgt;
global Im2;

[file,folder]= uigetfile( ...
       {'*.png'; ...
        '*.tiff'; ...
        '*.jpg'; ...
        '*.gif'}, ...
        'Enter ground truth');

filename2 =fullfile(folder,file);
set(handles.text3,'String', filename2);
if length(filename2)>4
```

```
    Xgt=imread(filename2);
    imshow(Xgt,'Parent',handles.axes2);
    Im2=1;
end
title(handles.axes2,sprintf('Ground Truth'),'fontsize',12);

% --- Executes on button press in radiobutton1.
function radiobutton1_Callback(hObject, eventdata, handles)
%=================================================================
%Sobel
%=================================================================
global F_Index;
    F_Index = 0;

    set(handles.radiobutton1,'Value',1);
    set(handles.radiobutton2,'Value',0);
    set(handles.radiobutton3,'Value',0);
    set(handles.radiobutton4,'Value',0);
    set(handles.radiobutton5,'Value',0);
    set(handles.radiobutton6,'Value',0);

% --- Executes on button press in radiobutton2.
function radiobutton2_Callback(hObject, eventdata, handles)
%=================================================================
%FoSobel-1
%=================================================================
global F_Index;
    F_Index = 1;

    set(handles.radiobutton2,'Value',1);
    set(handles.radiobutton1,'Value',0);
    set(handles.radiobutton3,'Value',0);
    set(handles.radiobutton4,'Value',0);
    set(handles.radiobutton5,'Value',0);
    set(handles.radiobutton6,'Value',0);

% --- Executes on button press in radiobutton3.
function radiobutton3_Callback(hObject, eventdata, handles)
%=================================================================
%FoSobel-2
%=================================================================
global F_Index;
    F_Index = 2;

    set(handles.radiobutton3,'Value',1);
    set(handles.radiobutton2,'Value',0);
    set(handles.radiobutton1,'Value',0);
    set(handles.radiobutton4,'Value',0);
    set(handles.radiobutton5,'Value',0);
    set(handles.radiobutton6,'Value',0);

% --- Executes on button press in radiobutton4.
function radiobutton4_Callback(hObject, eventdata, handles)
%=================================================================
%FoSobel-3
%=================================================================
global F_Index;
    F_Index = 3;

    set(handles.radiobutton4,'Value',1);
    set(handles.radiobutton2,'Value',0);
    set(handles.radiobutton3,'Value',0);
    set(handles.radiobutton1,'Value',0);
    set(handles.radiobutton5,'Value',0);
    set(handles.radiobutton6,'Value',0);


% --- Executes on button press in radiobutton5.
function radiobutton5_Callback(hObject, eventdata, handles)
%==================================================================
%FoSobel-4
%==================================================================
global F_Index;
    F_Index = 4;


    set(handles.radiobutton5,'Value',1);
    set(handles.radiobutton2,'Value',0);
    set(handles.radiobutton3,'Value',0);
    set(handles.radiobutton4,'Value',0);
    set(handles.radiobutton1,'Value',0);
    set(handles.radiobutton6,'Value',0);


% --- Executes on button press in radiobutton6.
function radiobutton6_Callback(hObject, eventdata, handles)
%==================================================================
%FoSobel-5
%==================================================================
global F_Index;
    F_Index = 5;

    set(handles.radiobutton6,'Value',1);
    set(handles.radiobutton2,'Value',0);
    set(handles.radiobutton3,'Value',0);
    set(handles.radiobutton4,'Value',0);
    set(handles.radiobutton5,'Value',0);
    set(handles.radiobutton1,'Value',0);



% --- Executes on slider movement.
function slider1_Callback(hObject, eventdata, handles)
% hObject    handle to slider1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'Value') returns position of slider
%        get(hObject,'Min') and get(hObject,'Max') to determine range
of slider
b=get(handles.slider1,'Value');
set(handles.edit1,'String',num2str(b));

% --- Executes during object creation, after setting all properties.
function slider1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to slider1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called
```

```
% Hint: slider controls usually have a light gray background.
if isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor',[.9 .9 .9]);
end


% --- Executes on slider movement.
function slider2_Callback(hObject, eventdata, handles)
% hObject    handle to slider2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'Value') returns position of slider
%        get(hObject,'Min') and get(hObject,'Max') to determine range
of slider

b=get(handles.slider2,'Value');
set(handles.edit2,'String',num2str(b));

% --- Executes during object creation, after setting all properties.
function slider2_CreateFcn(hObject, eventdata, handles)
% hObject    handle to slider2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: slider controls usually have a light gray background.
if isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor',[.9 .9 .9]);
end



function edit1_Callback(hObject, eventdata, handles)
% hObject    handle to edit1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit1 as text
%        str2double(get(hObject,'String')) returns contents of edit1
as a double
b=get(handles.edit1,'String');
set(handles.slider1,'Value',str2num(b));

% --- Executes during object creation, after setting all properties.
function edit1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
```

```
function edit2_Callback(hObject, eventdata, handles)
% hObject    handle to edit2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit2 as text
%        str2double(get(hObject,'String')) returns contents of edit2
as a double
b=get(handles.edit2,'String');
set(handles.slider2,'Value',str2num(b));


% --- Executes during object creation, after setting all properties.
function edit2_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Executes on button press in pushbutton4.
function pushbutton4_Callback(hObject, eventdata, handles)

%RUN Filter

global Im1;
global X;
global Im2;
global Xgt;
global F_Index;


if Im1==0
 uiwait(msgbox('Please, select Image'));
end
if Im2==0
 uiwait(msgbox('Please, select Image Ground truth'));
end
if Im1==1 && Im2==1
     p       = get(handles.slider2,'Value');
    switch  F_Index
        case 0
            [Mx, My]= dfsobdf1(0.0);
        case 1
            [Mx, My]= dfsobdf1(p);
        case 2
            [Mx, My]= dfsobdf2(p);
        case 3
            [Mx, My]= dfsobdf3(-p);
        case 4
            [Mx, My]= dfsobdf4(p);
        case 5
            [Mx, My]= dfsobdf5(p);
    end

    Scale    = get(handles.slider1,'Value');

    Ix       = imfilter(X,Mx,'replicate');
    Iy       = imfilter(X,My,'replicate');
    I3       = sqrt(double(Ix.^2+Iy.^2));
    Thresh   = sum(I3(:),'double') / numel(I3);
        I3 = I3>Scale*Thresh;
        I3 = (1-I3)*255;
    imshow(I3,'Parent',handles.axes3);
    title(handles.axes3,sprintf('Filtered Image'),'fontsize',12);


           I3  = I3 > 0;
           Gt1 = Xgt > 0;

           mse = sum(sum( (I3-double(Gt1)).^2 ) )/numel(I3);
          [TP,FP,FN,FM,PM,SDM,RDE,ME] =
score_of_evaluators(Gt1,I3,1,0.2);


          set(handles.edit3,'String',num2str(mse));
          set(handles.edit4,'String',num2str(SDM));
          set(handles.edit5,'String',num2str(RDE));
          set(handles.edit6,'String',num2str(ME));
          set(handles.edit7,'String',num2str(FM));
          set(handles.edit8,'String',num2str(PM));

          set(handles.edit10,'String',num2str(TP));
          set(handles.edit11,'String',num2str(FP));
          set(handles.edit12,'String',num2str(FN));

end

function edit3_Callback(hObject, eventdata, handles)
% hObject    handle to edit3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit3 as text
%        str2double(get(hObject,'String')) returns contents of edit3
as a double


% --- Executes during object creation, after setting all properties.
function edit3_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
```

```
function edit4_Callback(hObject, eventdata, handles)
% hObject    handle to edit4 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit4 as text
%        str2double(get(hObject,'String')) returns contents of edit4
as a double


% --- Executes during object creation, after setting all properties.
function edit4_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit4 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function edit5_Callback(hObject, eventdata, handles)
% hObject    handle to edit5 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit5 as text
%        str2double(get(hObject,'String')) returns contents of edit5
as a double


% --- Executes during object creation, after setting all properties.
function edit5_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit5 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function edit6_Callback(hObject, eventdata, handles)
% hObject    handle to edit6 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit6 as text
%        str2double(get(hObject,'String')) returns contents of edit6
as a double
```

```
% --- Executes during object creation, after setting all properties.
function edit6_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit6 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Executes on button press in pushbutton6.
function pushbutton6_Callback(hObject, eventdata, handles)
```

```
% Filter Analysis

global Im1;
global X;
global Im2;
global Xgt;
global F_Index;


if Im1==0
       uiwait(msgbox('Please, select Image'));
end
if Im2==0
       uiwait(msgbox('Please, select Image Ground truth'));
end
Scale   = get(handles.slider1,'Value');
Y1=[];Y2=[];Y3=[];Y4=[];Y5=[];Y6=[];Y7=[];
if Im1==1 && Im2==1

    for p=0:0.01:0.95

%==========================================================
% 1

%==========================================================
                 [Mx, My]= dfsobdf1(p);
                 Ix      = imfilter(X,Mx,'replicate');
                 Iy      = imfilter(X,My,'replicate');
                 I3      = sqrt(double(Ix.^2+Iy.^2));
                 Thresh  = sum(I3(:),'double') / numel(I3);
                 I3 = I3>Scale*Thresh;
                 I3 = (1-I3)*255;
                 I3  = I3 > 0;
                 Gt1 = Xgt > 0;
                 b1 = sum(sum( (I3-double(Gt1)).^2 ) )/numel(I3);
                [TP1,FP1,FN1,FM1,PM1,SDM1,RDE1,ME1] =
score_of_evaluators (Gt1,I3,1,0.2);
```

```
%===========================================================
%2

%====================+======================================
                 [Mx, My]= dfsobdf2(p);
                 Ix      = imfilter(X,Mx,'replicate');
                 Iy      = imfilter(X,My,'replicate');
                 I3      = sqrt(double(Ix.^2+Iy.^2));
                 Thresh  = sum(I3(:),'double') / numel(I3);
                 I3 = I3>Scale*Thresh;
                 I3 = (1-I3)*255;

                 I3  = I3 > 0;
                 Gt1 = Xgt > 0;
                 b2 = sum(sum( (I3-double(Gt1)).^2 ) )/numel(I3);
                [TP2,FP2,FN2,FM2,PM2,SDM2,RDE2,ME2] =
score_of_evaluators (Gt1,I3,1,0.2);




%===========================================================
%3

%===========================================================
                 [Mx, My]= dfsobdf3(-p);
                 Ix      = imfilter(X,Mx,'replicate');
                 Iy      = imfilter(X,My,'replicate');
                 I3      = sqrt(double(Ix.^2+Iy.^2));
                 Thresh  = sum(I3(:),'double') / numel(I3);
                 I3 = I3>Scale*Thresh;
                 I3 = (1-I3)*255;
                 I3  = I3 > 0;
                 Gt1 = Xgt > 0;
                 b3 = sum(sum( (I3-double(Gt1)).^2 ) )/numel(I3);
                [TP3,FP3,FN3,FM3,PM3,SDM3,RDE3,ME3] =
score_of_evaluators (Gt1,I3,1,0.2);

%===========================================================
%4

%===========================================================
                 [Mx, My]= dfsobdf4(-p);
                 Ix      = imfilter(X,Mx,'replicate');
                 Iy      = imfilter(X,My,'replicate');
                 I3      = sqrt(double(Ix.^2+Iy.^2));
                 Thresh  = sum(I3(:),'double') / numel(I3);
                 I3 = I3>Scale*Thresh;
                 I3 = (1-I3)*255;
                 I3  = I3 > 0;
                 Gt1 = Xgt > 0;
                 b4 = sum(sum( (I3-double(Gt1)).^2 ) )/numel(I3);
                [TP4,FP4,FN4,FM4,PM4,SDM4,RDE4,ME4] =
score_of_evaluators (Gt1,I3,1,0.2);

%===========================================================
%5
%============================================================
                  [Mx, My]= dfsobdf5(p);
                  Ix      = imfilter(X,Mx,'replicate');
                  Iy      = imfilter(X,My,'replicate');
                  I3      = sqrt(double(Ix.^2+Iy.^2));
                  Thresh  = sum(I3(:),'double') / numel(I3);
                  I3 = I3>Scale*Thresh;
                  I3 = (1-I3)*255;
                  I3  = I3 > 0;
                  Gt1 = Xgt > 0;
                  b5 = sum(sum( (I3-double(Gt1)).^2 ) )/numel(I3);
                 [TP5,FP5,FN5,FM5,PM5,SDM5,RDE5,ME5] =
score_of_evaluators (Gt1,I3,1,0.2);


%============================================================
                 if p==0

b3=b1;SDM3=SDM1;RDE3=RDE1;ME3=ME1;FM3=FM1;PM3=PM1;TP3=TP1;FP3=FP1;FN3
=FN1;

b4=b1;SDM4=SDM1;RDE4=RDE1;ME4=ME1;FM4=FM1;PM4=PM1;TP4=TP1;FP4=FP1;FN4
=FN1;

b5=b1;SDM5=SDM1;RDE5=RDE1;ME5=ME1;FM5=FM1;PM5=PM1;TP5=TP1;FP5=FP1;FN5
=FN1;
                 end
                  Y1=[Y1;p b1 b2 b3 b4 b5]  ;
                  Y2=[Y2;p SDM1 SDM2 SDM3 SDM4 SDM5]  ;
                  Y3=[Y3;p RDE1 RDE2 RDE3 RDE4 RDE5]  ;
                  Y4=[Y4;p ME1 ME2 ME3 ME4 ME5]  ;
                  Y5=[Y5;p FM1 FM2 FM3 FM4 FM5];
                  Y6=[Y6;p PM1 PM2 PM3 PM4 PM5];
                  Y7=[Y7;p TP1 FP1 FN1 TP2 FP2 FN2 TP3 FP3 FN3 TP4 FP4
FN4 TP5 FP5 FN5];

%=====================================================================
==
                 %Y5=[Y5;p s2];
    end

%=====================================================================
==
%Mean Square Error
       figure,

plot(Y1(:,1),Y1(:,2),Y1(:,1),Y1(:,3),Y1(:,1),Y1(:,4),Y1(:,1),Y1(:,5),
Y1(:,1),Y1(:,6));
       legend('Operater 1','Operater 2 ','Operater 3','Operater
4','Operater 5') ;
       ylabel(' Mean Square Error'); xlabel('order (p)');grid;

%=====================================================================
==
% Symmetric Distance
       figure,
plot(Y2(:,1),Y2(:,2),Y2(:,1),Y2(:,3),Y2(:,1),Y2(:,4),Y2(:,1),Y2(:,5),
Y2(:,1),Y2(:,6));
       legend('Operater 1','Operater 2 ','Operater 3','Operater 
4','Operater 5') ;
       ylabel(' Symmetric Distance '); xlabel('order (p)');grid;

%======================================================================
==
% Relative Distance Error
       figure,

plot(Y3(:,1),Y3(:,2),Y3(:,1),Y3(:,3),Y3(:,1),Y3(:,4),Y3(:,1),Y3(:,5),
Y3(:,1),Y3(:,6));
       legend('Operater 1','Operater 2 ','Operater 3','Operater 
4','Operater 5') ;
       ylabel(' Relative Distance Error'); xlabel('order (p)');grid


%======================================================================
==
% Misclassification Error
       figure,

plot(Y4(:,1),Y4(:,2),Y4(:,1),Y4(:,3),Y4(:,1),Y4(:,4),Y4(:,1),Y4(:,5),
Y4(:,1),Y4(:,6));
       legend('Operater 1','Operater 2 ','Operater 3','Operater 
4','Operater 5') ;
       ylabel(' Misclassification Error'); xlabel('order (p)');grid;

%======================================================================
==
% FM Complement
       figure,

plot(Y5(:,1),Y5(:,2),Y5(:,1),Y5(:,3),Y5(:,1),Y5(:,4),Y5(:,1),Y5(:,5),
Y5(:,1),Y5(:,6));
       legend('Operater 1','Operater 2 ','Operater 3','Operater 
4','Operater 5') ;
       ylabel(' FM Complement  '); xlabel('order (p)');grid;

%======================================================================
==
       figure,

plot(Y6(:,1),Y6(:,2),Y6(:,1),Y6(:,3),Y6(:,1),Y6(:,4),Y6(:,1),Y6(:,5),
Y6(:,1),Y6(:,6));
       legend('Operater 1','Operater 2 ','Operater 3','Operater 
4','Operater 5') ;
       ylabel(' PM Complement '); xlabel('order (p)');grid;


%======================================================================
==
% True Positive

       figure,

plot(Y7(:,1),Y7(:,2),Y7(:,1),Y7(:,5),Y7(:,1),Y7(:,8),Y7(:,1),Y7(:,11)
,Y7(:,1),Y7(:,14));
```

```
       legend('Operater 1','Operater 2 ','Operater 3','Operater
4','Operater 5') ;
       ylabel(' True Positive '); xlabel('order (p)');grid;

%=====================================================================
==
% False Positive
       figure,

plot(Y7(:,1),Y7(:,3),Y7(:,1),Y7(:,6),Y7(:,1),Y7(:,9),Y7(:,1),Y7(:,12)
,Y7(:,1),Y7(:,15));
       legend('Operater 1','Operater 2 ','Operater 3','Operater
4','Operater 5') ;
       ylabel(' False Positive '); xlabel('order (p)');grid;

%=====================================================================
==
% False Negative

       figure,

plot(Y7(:,1),Y7(:,4),Y7(:,1),Y7(:,7),Y7(:,1),Y7(:,10),Y7(:,1),Y7(:,13
),Y7(:,1),Y7(:,16));
       legend('Operater 1','Operater 2 ','Operater 3','Operater
4','Operater 5') ;
       ylabel(' False Negative '); xlabel('order (p)');grid;


%=====================================================================
==

% Excel File

       xlhead={'Order','FoS-Yi_FeiPU-1','FoS-Yi_FeiPU-2','FoS-
Yi_FeiPU-3','FoS-Yi_FeiPU-4','FoS-Yi_FeiPU-5'};
       Mtrics={'Mean Square Error','Symmetric Distance','Relative
Distance Error','Misclassification Error','F Measure','Performance
Measure','(TP, FP, FN)'};
       name='Result1.xls';
       xlswrite('Result1.xls',Mtrics(1),1,'C4');
       xlswrite('Result1.xls',xlhead,1,'D4');
       xlswrite('Result1.xls',Y1,1,'D5');

%=====================================================================
==
       xlswrite('Result1.xls',Mtrics(2),1,'C104');
       xlswrite('Result1.xls',xlhead,1,'D104');
       xlswrite('Result1.xls',Y2,1,'D105');

%=====================================================================
==
       xlswrite('Result1.xls',Mtrics(3),1,'C204');
       xlswrite('Result1.xls',xlhead,1,'D204');
       xlswrite('Result1.xls',Y3,1,'D205');

%=====================================================================
==
       xlswrite('Result1.xls',Mtrics(4),1,'C304');
       xlswrite('Result1.xls',xlhead,1,'D304');
       xlswrite('Result1.xls',Y4,1,'D305');
```

```
%======================================================================
==
       xlswrite('Result1.xls',Mtrics(5),1,'C404');
       xlswrite('Result1.xls',xlhead,1,'D404');
       xlswrite('Result1.xls',Y5,1,'D405');

%======================================================================
==
       xlswrite('Result1.xls',Mtrics(6),1,'C504');
       xlswrite('Result1.xls',xlhead,1,'D504');
       xlswrite('Result1.xls',Y6,1,'D505');

%======================================================================
==
       xlswrite('Result1.xls',Mtrics(7),1,'C604');
  %     xlswrite('Result1.xls',xlhead,1,'D604');
       xlswrite('Result1.xls',Y7,1,'D605');

%======================================================================
==

       msgbox('Results is stored in Excel File');

end



function edit7_Callback(hObject, eventdata, handles)
% hObject    handle to edit7 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit7 as text
%        str2double(get(hObject,'String')) returns contents of edit7
as a double


% --- Executes during object creation, after setting all properties.
function edit7_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit7 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function edit8_Callback(hObject, eventdata, handles)
% hObject    handle to edit8 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit8 as text
%        str2double(get(hObject,'String')) returns contents of edit8
as a double


% --- Executes during object creation, after setting all properties.
function edit8_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit8 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function edit9_Callback(hObject, eventdata, handles)
% hObject    handle to edit9 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit9 as text
%        str2double(get(hObject,'String')) returns contents of edit9
as a double


% --- Executes during object creation, after setting all properties.
function edit9_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit9 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function edit10_Callback(hObject, eventdata, handles)
% hObject    handle to edit10 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit10 as text
%        str2double(get(hObject,'String')) returns contents of edit10
as a double


% --- Executes during object creation, after setting all properties.
function edit10_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit10 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function edit11_Callback(hObject, eventdata, handles)
% hObject    handle to edit11 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit11 as text
%        str2double(get(hObject,'String')) returns contents of edit11
as a double


% --- Executes during object creation, after setting all properties.
function edit11_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit11 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function edit12_Callback(hObject, eventdata, handles)
% hObject    handle to edit12 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit12 as text
%        str2double(get(hObject,'String')) returns contents of edit12
as a double


% --- Executes during object creation, after setting all properties.
function edit12_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit12 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
```

```
% --- Executes on button press in pushbutton8.
function pushbutton8_Callback(hObject, eventdata, handles)



% View Edges

global Im1;
global X;
global Im2;
global Xgt;
global F_Index;
warning off;
clc;
if Im1==0
       uiwait(msgbox('Please, select Image'));
end
if Im2==0
       uiwait(msgbox('Please, select Image Ground truth'));
end
Scale   = get(handles.slider1,'Value');
Y=[];

if Im1==1 && Im2==1

    prompt={'Enter order of differentiation - initial vale:','Enter
step value:','Enter final value:'};
    name='Input range of differentiation';
    numlines=1;
    defaultanswer={'0','0.1','0.9'};
    options.Resize='on';
    options.WindowStyle='normal';
    options.Interpreter='tex';
    answer=inputdlg(prompt,name,numlines,defaultanswer,options);
    oder_I=str2num(answer{1});
    oder_S=str2num(answer{2});
    oder_F=str2num(answer{3});

    for p=oder_I:oder_S:oder_F


%============================================================
                 %                    1

%============================================================
                 [Mx, My]= dfsobdf1(p);
                 Ix      = imfilter(X,Mx,'replicate');
                 Iy      = imfilter(X,My,'replicate');
                 I3      = sqrt(double(Ix.^2+Iy.^2));
                 Thresh  = sum(I3(:),'double') / numel(I3);
                 I3 = I3>Scale*Thresh;
                 I1 = (1-I3)*255;

%=============================================================
                %                     2

%=====================+=======================================
                 [Mx, My]= dfsobdf2(p);
                 Ix      = imfilter(X,Mx,'replicate');
                 Iy      = imfilter(X,My,'replicate');
                 I3      = sqrt(double(Ix.^2+Iy.^2));
                 Thresh  = sum(I3(:),'double') / numel(I3);
                 I3 = I3>Scale*Thresh;
                 I2 = (1-I3)*255;


%============================================================
                %                     3

%============================================================
                 [Mx, My]= dfsobdf3(-p);
                 Ix      = imfilter(X,Mx,'replicate');
                 Iy      = imfilter(X,My,'replicate');
                 I3      = sqrt(double(Ix.^2+Iy.^2));
                 Thresh  = sum(I3(:),'double') / numel(I3);
                 I3 = I3>Scale*Thresh;
                 I33 = (1-I3)*255;

%============================================================
                %                     4

%============================================================
                 [Mx, My]= dfsobdf4(-p);
                 Ix      = imfilter(X,Mx,'replicate');
                 Iy      = imfilter(X,My,'replicate');
                 I3      = sqrt(double(Ix.^2+Iy.^2));
                 Thresh  = sum(I3(:),'double') / numel(I3);
                 I3 = I3>Scale*Thresh;
                 I4 = (1-I3)*255;

%============================================================
                %                     5

%============================================================
                 [Mx, My]= dfsobdf5(p);
                 Ix      = imfilter(X,Mx,'replicate');
                 Iy      = imfilter(X,My,'replicate');
                 I3      = sqrt(double(Ix.^2+Iy.^2));
                 Thresh  = sum(I3(:),'double') / numel(I3);
                 I3 = I3>Scale*Thresh;
                 I5 = (1-I3)*255;

                 if p==0.0
                     x=[uint8(Xgt);
uint8(I1);uint8(I1);uint8(I1);uint8(I1);uint8(I1)];
                     Y=[Y x];
                 else

x=[uint8(Xgt);uint8(I1);uint8(I2);uint8(I3);uint8(I4);uint8(I5)];
                     Y=[Y x];
                end
    end
    figure, montage(Y),title(' Ground Truth First row FoS-1  :Second
row FoS-2   :Third row FoS-Yi_FeiPU-3    :Fourth row FoS-4    :Fifth
row FoS-5');
end
```

البحوث المنشورة

[1] Eng. Ibtisam Edress, Dr. Emad A. Al-Sabawi, and Dr. Majid Dherar Younus, “Design of Fractional-order Sobel Filters for Edge Detections,” 1st International Ninevah Conference on Engineering and Technology(INCET2021), Mosul,Goverence,Iraq for the period from (5-6) April 2021. The conference will publish the papers in IOP Conference Series: Materials Science and Engineering, Scopus indexed.

الخلاصــة

يعد اكتشاف الحواف أحد مشكلات البحث الحيوية وخطوة رئيسية مهمة جدًا نحو استخلاص ميزات الصورة وذلك لتجزئتها والتعرف على انماطها. يعتمد على مشتقات الأعداد الصحيحة لاكتشاف حواف الصورة ، خاصة المشتقة من الدرجة الأولى أو الثانية. مع ظهور حساب التفاضل والتكامل الكسري ، تمت اعادة النظرفي اكتشاف الحوافي باستخدام المشتقه الكسرية. تم في هذا العمل اقتراح طريقه للكشف عن الحوافي في الفيديو استنادًا إلى خمسة مرشحات كسرية المرتبة لـ Sobel بناءا على معاملات الحساب الكسري لـ Yi_Fei-1 إلى Yi_Fei-5.

تم تطوير واجهة رسومية للمستخدم (GUI) في (Matlab 2017b) لتحليل المرشحات كسرية المرتبة المقترحة لـ Sobel والكلاسيكية. تعتبر دقة الحوافي احدى اهم التحديات لـ أي كاشف حواف ؛ لهذا تم إجراء تقييم خاضع للإشراف بين المرشحات المقترحة والمرشح الكلاسيكي باستخدام Mean Square Error و Symmetric Distance Measure و Misclassification Error Complemented و Distance Error Relative Performance Measure و Complemented F Measure . حيث تم استخدام العديد من الصور مع حوافها الحقيقية في التقييم. أظهرت النتائج أن مرشح Sobel كسري المرتبة المعتمد على Yi_Fei-2 يتفوق على المرشح التقليدي والمرشحات المقترحة الأخرى.

تم تنفيذ المرشح الكسري المرتبة لـ Sobel المعتمد على Yi_Fei-2 باستخدام مصفوفة بوابات منطقية مبرمجة حقليا (FPGA) حيث استخدمت برمجيات شركة زايلينكس (Xilinx Vivado 2018.2) بالتكامل مع (Matlab 2017b). تم استخدام محقق الـ HDL من خلال (FPGA in Loop) و(Co-simulation) في بناء مصفوفة بوابات منطقية مبرمجة حقليا على لوحة Artix-7 NEXYS 4. وجد ان (root mean square error) بين محاكاة الكيان المادي والمحاكاة البرمجية هو (0.002).

اقرار لجنة المناقشة

نشهد بأننا أعضاء لجنة التقويم والمناقشة قد اطلعنا على هذه الرسالة الموسومة (بناء مصفوفة بوابات مبرمجه حقليا لكشف حوافي الفيديو بالاعتماد على التفاضل كسري المرتبة) وناقشنا الطالبة (ابتسام ادريس كنعان) في محتوياتها وفيما له علاقة بها بتاريخ / / 2021 وقد وجدناها جدير بنيل شهادة الماجستير–علوم في اختصاص هندسة الحاسوب والمعلوماتية.

التوقيع:	التوقيع:
رئيس اللجنة:	عضو اللجنة:
د.محمد حازم الجماس	د.بسمة محمدكمال يونس
التاريخ: / /2021	التاريخ: / /2021

التوقيع:

عضو اللجنة:

د.سنان حسام مهدي

التاريخ: / /2021

التوقيع:	التوقيع:
عضو اللجنة (المشرف) :	عضو اللجنة (المشرف):
د.عماد عطيه خلف	د.ماجد ضرار يونس
التاريخ: / /2021	التاريخ: / /2021

قرار مجلس الكلية

اجتمع مجلس كلية هندسة الالكترونيات بجلسته المنعقدة بتاريخ : / / 2021 وقرر المجلس منح الطالب شهادة الماجستير علوم في اختصاص هندسة الحاسوب والمعلوماتية.

مقرر المجلس: د.	رئيس مجلس الكلية:
التاريخ: / / 2021	التاريخ: / / 2021

إقرار المشرف

اشهد بان الرسالة الموسومة ب " **بناء مصفوفة بوابات مبرمجه حقليا لكشف حوافي الفيديو بالاعتماد على التفاضل كسري المرتبة"** تمت تحت اشرافي وهي جزء من متطلبات نيل شهادة الماجستير في هندسة الحاسوب والمعلوماتية

التوقيع:
المشرف : د. عماد عطيه خلف
التاريخ: /   / 2021

التوقيع:
المشرف : د. ماجد ضرار يونس
التاريخ: /   / 2021

إقرار المقيم اللغوي

اشهد باني قمت بمراجعة الرسالة الموسومة ب " **بناء مصفوفة بوابات مبرمجه حقليا لكشف حوافي الفيديو بالاعتماد على التفاضل كسري المرتبة** " من الناحية اللغوية وتصحيح ما ورد فيها من أخطاء لغوية وتعبيرية وبذلك أصبحت الرسالة مؤهلة للمناقشة بقدر تعلق الامر بسلامة الأسلوب وصحة التعبير.

التوقيع:
المقوم اللغوي:
التاريخ: /   / 2021

إقرار رئيس لجنة الدراسات العليا

بناء على التوصيات المقدمة من قبل المشرف والمقوم اللغوي أرشح هذه الرسالة للمناقشة.
التوقيع:
الاسم: أ . م . معن أحمد شحاذة العدواني
التاريخ: /   / 2021

<u>إقرار رئيس القسم</u>

بناء على التوصيات المقدمة من قبل المشرف والمقوم اللغوي ورئيس لجنة الدراسات العليا أرشح هذه الرسالة للمناقشة.

التوقيع:

الاسم: أ. م. معن أحمد شحاذة العدواني

التاريخ: /   / 2021

بناء مصفوفة بوابات مبرمجه حقليا لكشف حوافي الفيديو بالاعتماد على التفاضل كسري المرتبة

رسالة تقدمت بها

ابتسام ادريس كنعان سليمان

إلى

مجلس كلية هندسة الالكترونيات

جامعة نينوى

كجزء من متطلبات نيل شهادة الماجستير

في

هندسة الحاسوب والمعلوماتية

بإشراف

د. عماد عطيه خلف **د. ماجد ضرار يونس**

١٤٤٢ هـ ٢٠٢١م

جامعة نينوى

كلية هندسة الالكترونيات

قسم هندسة الحاسوب والمعلوماتية

بناء مصفوفة بوابات مبرمجة حقليا لكشف حوافي الفيديو بالاعتماد على التفاضل كسري المرتبة

ابتسام ادريس كنعان سليمان

رسالة ماجستير علوم في هندسة الحاسوب والمعلوماتية

بإشراف

د. عماد عطيه خلف   د. ماجد ضرار يونس

١٤٤٢ هـ   ٢٠٢١ م